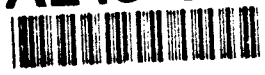


(2)

MEMORANDUM REPORT BRL-MR-3954

BRL

MODULAR UNIX®-BASED VULNERABILITY ESTIMATION SUITE (MUVES) ANALYST'S GUIDE

PHILLIP J. HANES
SCOTT L. HENRY
GARY S. MOSS
KAREN R. MURRAY
WENDY A. WINNER

DECEMBER 1991

U.S. ARMY LABORATORY COMMAND

BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

NOTICES

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1991	FINAL JUL 88 - OCT 91

4. TITLE AND SUBTITLE
Modular UNIX®-based Vulnerability Estimation Suite (MUVES) Analyst's Guide

5. FUNDING NUMBERS
IL162618AH80

6. AUTHOR(S)
Phillip J. Hanes, Scott L. Henry, Gary S. Moss, Karen R. Murray, Wendy A. Winner

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US ARMY BALLISTIC RESEARCH LABORATORY
ATTN: SLCBR-DD-T
ABERDEEN PROVING GROUND, MD 21005-5066

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
BRL-MR-3954

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Assessments of a target's susceptibility to damage in an encounter with a threat (vulnerability) and the capability of weapon systems to inflict damage on targets (lethality) have been conducted at the Ballistic Research Laboratory (BRL), Aberdeen Proving Ground, MD, for over forty years. Research and data from test firings have been incorporated in a series of computer models that predict the results of target-threat interactions. Prior to the Modular UNIX®-based Vulnerability Estimation Suite (MUVES), many computer programs for predicting vulnerability/lethality assessments had evolved with their own blend of specially tailored algorithms and with their own code maintenance requirements. Even though these programs functionally employed many of the same algorithms, code was not directly shared, and it was not uncommon to find vulnerability analysts using slightly different versions of the same code. In 1985, BRL's Vulnerability/Lethality Division (VLD) initiated discussions to develop an *integrated* software system for performing vulnerability/lethality assessments. These efforts have resulted in MUVES, a comprehensive software package for vulnerability/lethality analysis. MUVES is written primarily in C for the UNIX operating system environment. MUVES is a flexible platform able to accommodate existing vulnerability methodologies as well as future vulnerability applications.

®UNIX is a registered trademark of AT&T.

14. SUBJECT TERMS			15. NUMBER OF PAGES
computerized simulation models	geometric modeling		167
vulnerability analysis	tanks (combat vehicles)		16. PRICE CODE
projectiles	user manuals		
17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

INTENTIONALLY LEFT BLANK.

Page

TABLE OF CONTENTS

PREFACE xiii

ACKNOWLEDGEMENTS xv

1. Introduction 1

1.1 Historical Perspective 1

1.2 Project Goals 2

1.3 Project Description 4

1.4 Advantages 7

1.5 Configuration Management 8

1.6 State of Development 9

1.7 Additional Documentation and Training 9

1.8 Software Distribution 10

2. System Design 11

2.1 Structure 11

2.2 Approximation Methods 13

2.3 Processes 14

2.4 MUVES Output Files 15

2.4.1 Final Results Files 15

2.4.2 Intermediate Results Files 16

3. Analyst Supplied Input Files 17

3.1 Input File Syntax 17

3.2 Threat Information 21

3.2.1 Initial File 22

3.2.2 Supplemental Threat Files 23

3.3 Targets 25

3.3.1 Ray-tracing host file 25

3.3.2 Target Description 26

3.3.3 Region Map File 26

3.3.4 Geometry File 27

3.3.5 View File 28

3.3.6 Reusable Ray File 34

3.3.7 Component Properties File 35

3.3.8 Component Category Map File 36

3.3.9 Damage Assessment Expression File 38

3.3.10 System Definition File 41

Page

3.3.11 Damage Evaluation Selection File 42
3.4 Underlying Approximation Method Inputs 43
3.4.1 Interaction Modules Inputs 44
3.4.2 Evaluation Modules Inputs 44
3.5 File Formats summarized using Backus-Naur Form (Optional) 45
3.5.1 Notation (Optional) 45
3.5.2 Backus-Naur notation for "initial" files (Optional) 47
3.5.3 Backus-Naur notation for ray-tracing "host" files 48
3.5.4 Backus-Naur notation for MGED region map files 48
3.5.5 Backus-Naur notation for "geometry" files 49
3.5.6 Backus-Naur notation for component properties files 49
3.5.7 Backus-Naur notation for damage assessment expressions and system definitions 49
3.5.8 Backus-Naur notation for damage evaluation selection files 52
4. User Interface 53
4.1 User Interface – The Basics 53
4.1.1 Using the Menus 54
4.1.2 Prompted Input 58
4.1.3 Scrolling the Text Region 59
4.1.4 MUVES Start up File 62
4.2 The Menu Hierarchy 64
4.2.1 Analysis 64
4.2.1.1 Analysis Configuration – Sessions 66
4.2.1.2 Analysis Configuration – Input Selection 68
4.2.1.2.1 Input File and Input Parameter Selection 68
4.2.1.2.2 Loading a Command File 73
4.2.1.2.3 Import – Moving Input Files into the MUVES File Hierarchy 73
4.2.1.2.4 Export – Moving Input Files out of the MUVES File Hierarchy 75
4.2.1.3 Configuration and Analysis 76
4.2.1.4 Access List Configuration 76
4.2.2 Results Interpretation 77
4.2.3 File Administration 79
4.2.3.1 Removing Project Files – Individual Files 79
4.2.3.2 Removing Project Files – Entire Directories 82
4.2.3.3 Removing Entire Projects 82

Page

4.2.3.4 Archiving Project Files 82
4.2.3.5 Administrating Project Lock Files 83
5. Postprocessors 84
5.1 Postprocessors for Intermediate Results Files 85
5.1.1 ASCII conversion 85
5.2 Postprocessors for Final Results Files 87
5.2.1 Compartment-Style Probability Outputs 87
5.2.2 Color Silhouettes 92
5.2.3 Cell-by-Cell Results 95
References 97
Appendix A: Online Manual Pages 99
Glossary 121
List of Symbols and Abbreviations 127
Distribution List 131

INTENTIONALLY LEFT BLANK.

Page

LIST OF FIGURES

Figure 1. MUVES project goals 2
Figure 2. Top-level MUVES description 8
Figure 3. MUVES Analysis Principal Data Flows 11
Figure 4. A sample one-dimensional table file 20
Figure 5. A sample two-dimensional table file 20
Figure 6. A sample table file containing multiple tables 21
Figure 7. Generic "initial" file 22
Figure 8. A sample kinetic-energy "range" file 23
Figure 9. A sample kinetic-energy "perf" file 24
Figure 10. A sample "host" file using the local host 26
Figure 11. A sample "host" file using a remote host 26
Figure 12. A sample MGED region map file 27
Figure 13. A sample "geometry" file 28
Figure 14. Example of Shot Grouping 31
Figure 15. Center of Grid 32
Figure 16. Standard View file for VLD Analyses 34
Figure 17. A sample component properties file 36
Figure 18. A sample component category map file 37
Figure 19. Unary and binary operators 39
Figure 20. A sample damage assessment expression file 40
Figure 21. A sample system definition file 42
Figure 22. A sample damage evaluation selection file 43
Figure 23. Backus-Naur notation for "initial" files 47
Figure 24. Backus-Naur notation for ray-tracing "host" files 48
Figure 25. Backus-Naur notation for MGED region map files 48
Figure 26. Backus-Naur notation for "geometry" files 49
Figure 27. Backus-Naur notation for component properties files 49
Figure 28. Unary and binary operators 50
Figure 29. Backus-Naur notation common to damage assessment expressions and system definitions 51
Figure 30. Backus-Naur notation specific to damage assessment expression files 52
Figure 31. Backus-Naur notation specific to system definition files 52
Figure 32. Backus-Naur notation for damage evaluation selection files 52

Page

Figure 33 Terminal Window after **muvee** has been Started 54
Figure 34 Scrolled Menus 55
Figure 35 Scrolled Menu – Top and Bottom 56
Figure 36 Setup Options Menu 57
Figure 37 Analysis Management Menu 57
Figure 38 File Browser Menu 57
Figure 39 Key Bindings Menu 58
Figure 40 Exit **muvee** Confirmation Prompt 58
Figure 41 Scrolling Region 61
Figure 42 Sample ".muveerc" File 62
Figure 43 Main Menu 64
Figure 44 Analysis Processing 64
Figure 45 Project Selection Submenu 64
Figure 46 List of Projects Submenu 65
Figure 47 Analysis Configuration Submenu 66
Figure 48 End Session Confirmation Prompt 66
Figure 49 Session Files Submenu 66
Figure 50 Load Session File Submenu 66
Figure 51 Run Menu 66
Figure 52 Reset Configuration Prompt 67
Figure 53 Export Submenu 67
Figure 54 Inputs Submenu 68
Figure 55 Input Parameter Selection Menu 69
Figure 56 Queue Submenu 69
Figure 57 Assessment Expression Environment/Mission Menu 70
Figure 58 View Queue Submenu 71
Figure 59 View Pattern Submenu 71
Figure 60 Shot Grouping Submenu 71
Figure 61 Shot Coordinate Menu 72
Figure 62 Setting Reusable Rays Submenu 72
Figure 63 Import Files Submenu 74
Figure 64 File Types to Import 74
Figure 65 Entire Directory or Individual File Menu 74
Figure 66 Import File Prompt 74
Figure 67 New File Prompt 75
Figure 68 Access List File (example) 76

Page

Figure 69. Access List Configuration Submenu 77
Figure 70. Type of Results Submenu 78
Figure 71. Current Project's Intermediate and Final Results Selection Submenu 78
Figure 72. Intermediate Results Postprocessors Submenu 78
Figure 73. Final Results Postprocessors Submenu 79
Figure 74. File Administration Submenu 79
Figure 75. Removal of Files Submenu 79
Figure 76. Remove Individual File Submenu 80
Figure 77. Edit List of Files Submenu 80
Figure 78. File Removal Category Submenu 80
Figure 79. Inputs Submenu 80
Figure 80. Sample Threat File Removal Menu 81
Figure 81. Remove Directory Prompt 82
Figure 82. Archival of Files Submenu 82
Figure 83. Sample from an Intermediate Results File 86
Figure 84. Sample View Information File Entry 89
Figure 85. Summary Table Sample 90
Figure 86. Beginning of an IUA File 91
Figure 87. View Average File 92
Figure 88. `colorail` Black and White Image 94
Figure 89. Sample cellxcell output 96

INTENTIONALLY LEFT BLANK.

Page

LIST OF TABLES

TABLE 1. Interpolation Table Flags 19
TABLE 2. Shot Definition Keywords 29
TABLE 3. Valid Unit Names 30
TABLE 4. Menu Manipulation Commands and their Default Bindings. 55
TABLE 5. Pop-up Menu Commands and their Default Bindings. 57
TABLE 6. Prompt Editing Commands 59
TABLE 7. Scrolling Region Commands and their Default Bindings. 60
TABLE 8. Start up File Directives 63
TABLE 9. Input File Types 81
TABLE 10. Valid Unit Names 85
TABLE 11. Remaining Functionality Color Conversion Key 93

INTENTIONALLY LEFT BLANK.

PREFACE

Assessments of a target's susceptibility to damage in an encounter with a threat (vulnerability) and the capability of weapon systems to inflict damage on targets (lethality) have been conducted at the Ballistic Research Laboratory (BRL), Aberdeen Proving Ground, MD, for over forty years. Through the years, research and data from test firings have been incorporated in a series of computer models that predict the results of target-threat interactions. Prior to the Modular UNIX-based Vulnerability Estimation Suite (MUVES), many computer programs for predicting vulnerability/lethality assessments had evolved with their own blend of specially tailored algorithms and with their own code maintenance requirements. Even though these programs functionally employed many of the same algorithms, code was not directly shared, and it was not uncommon to find vulnerability analysts using slightly different versions of the same code. In 1985, BRL's Vulnerability/Lethality Division (VLD) initiated discussions to develop an *integrated* software system for performing vulnerability/lethality assessments. These efforts have resulted in MUVES, a comprehensive software package for vulnerability/lethality analysis. MUVES is written primarily in C for the UNIX operating system environment. MUVES is a flexible platform able to accommodate existing vulnerability methodologies as well as future vulnerability applications.

This guide describes MUVES at the level a vulnerability analyst needs to successfully perform vulnerability/lethality studies. Appendices describe the currently supported algorithm packages and are published under separate cover. As existing vulnerability algorithms are incorporated into MUVES and new algorithms are developed, additional documentation will be made available.

INTENTIONALLY LEFT BLANK.

ACKNOWLEDGEMENTS

MUVES has been designed and implemented by a group of personnel from the Vulnerability Methodology Branch of the Vulnerability/Lethality Division. In alphabetical order, members of the MUVES Team presently include:

Douglas A. Gwyn
Phillip J. Hanes
Scott L. Henry
Gary S. Moss
Karen R. Murray
Jill H. Smith
Wendy A. Winner

The authors of this report thank Douglas A. Gwyn for his diligent efforts in reviewing preliminary versions of this guide and providing numerous useful suggestions which were incorporated into the final report. We also thank Charles J. Huenke Jr. and Lawrence W. Wilson from the Ground Systems Branch of the Vulnerability/Lethality Division for their comments and suggestions. Finally, we thank Robert D. Wilson from the Ground Systems Branch of the Vulnerability/Lethality Division and Eric G. Heilman from the Weapon Systems Technology Branch of the System Engineering and Concepts Analysis Division for their thorough review of this report.

INTENTIONALLY LEFT BLANK.

1. Introduction

1.1 Historical Perspective

The Ballistic Research Laboratory (BRL), Aberdeen Proving Ground, MD, assesses weapon systems' susceptibilities to damage in encounters with threats (vulnerabilities). The laboratory also studies the capabilities of weapon systems to inflict damage on targets (lethalities).

Vulnerability/lethality research includes data collected from laboratory test firings and full-scale field testing as well as the series of computer models which predict results of target/threat interactions [1]. These models have progressed from compartment-level models with lumped-parameter estimates for weapon system damage [2] to point-burst models that can evaluate behind-armor debris and threat/debris interactions with critical interior components [3,4]. More recently, a stochastic point-burst model has been developed to predict damage and its associated probabilities at the component level [5]. This model provides detailed modeling support for live-fire testing. Using a hierarchy of models, the BRL has an iterative process for improving both the vulnerability/lethality discipline and the designs of fielded systems.

Production use of simulation models for more than thirty years has necessitated continual incorporation of state-of-the-art research (*e.g.*, algorithm improvements for weapon system technologies) and the periodic introduction of new models based on the work of earlier models. Once created, these models have tended to evolve with their own blend of specially tailored algorithms and methodologies. Computer programs implementing the three primary types of Vulnerability/Lethality Division (VLD) models, *i.e.*, the compartment, point-burst, and stochastic point-burst models, have employed many of the same algorithms but have not shared code. Consequently, each of these models required individual support.

In 1985, BRL's VLD initiated internal discussions to develop an *integrated* software code for performing vulnerability/lethality assessments to both overcome these problems and to advance the state-of-the-art. These efforts have culminated in the Modular UNIX®-based Vulnerability Estimation Suite (MUVES), a software package written in the C programming language for UNIX operating systems. MUVES has been designed for the study of target-threat interactions in an environment capable of accommodating existing vulnerability methodologies as well as providing flexibility to support newly devised requirements [6].

This document introduces vulnerability/lethality terminology and methodologies, describes MUVES, and supplies information that both new and experienced vulnerability analysts will need for performing MUVES analyses.

® UNIX is a registered trademark of AT&T.

1.2 Project Goals

- Provide current vulnerability analysis capabilities in the UNIX environment
- Control production code design and maintenance
- Minimize code redundancy
- Minimize analyst's programming
- Provide flexible user-specified program interaction
- Facilitate extension of production code for:
  - — New situations
  - — New methods of analysis
  - — Experimental applications
- Use BRL's Multi-device Graphics Editor and ray-tracing packages for solid modeling and target geometry interrogation
- Provide user-friendly interface

Figure 1. MUVES project goals

Figure 1 summarizes the goals of the MUVES effort [6]. MUVES currently incorporates vulnerability analysis capabilities available under the compartment-level model. As part of coding this model, general-purpose software packages (*e.g.*, error-handling routines, doubly-linked lists, memory management) were also written and will be used for other vulnerability/lethality models.

Selecting the computing environment for this code was one of several critical decisions. The UNIX operating system was selected for several reasons. Many classes of computers are now available with some variant of the UNIX operating system: single-user graphics workstations, minicomputers, and supercomputers. Given the popularity, availability, and capabilities of the UNIX operating system, acquisitions of UNIX-based systems are expected to continue for federal scientific research and development applications.

Given the MUVES code can be made available for classes of computer systems with the UNIX operating system,* an analyst can select the most appropriate or convenient computer system(s) for analysis tasks.

* For each release, more specific information on the UNIX environment requirements will be provided.

For example, ray tracing a geometric target description is computationally intensive but integral to vulnerability computations. The MUVES executable can be run on a local machine where the local host performs all tasks, or the executable can be run such that the local host performs the vulnerability tasks and another network-accessible host performs ray-tracing tasks.

The VLD has set a goal to improve the configuration management of its vulnerability codes. This goal will eliminate customized implementations of a particular vulnerability/lethality model and will insure consistent analysis results across the division. This is important since the division anticipates long-term production use of its models.

Software engineering design techniques were used to fulfill several of the MUVES goals (*e.g.*, long-term quality of the computer code). These techniques have been developed by the computer science community to control code design and maintenance [7,8]. By outlining basic software functions and inter-relationships, these techniques can minimize code redundancy and can be used to more easily assess the impact of new production code. Using these techniques, the flow of data within the MUVES code was diagramed.

In previous vulnerability codes, a particular penetration algorithm may have been independently implemented in two or more versions of the same code. Consequently, a change to the algorithm (*e.g.*, an improvement or a replacement) meant that each instance of the algorithm in every version of the code needed to be updated. A goal of MUVES is to have only one module written and maintained to accomplish a particular calculation. This avoids having multiple implementations of the same algorithm and insures analysts have immediate access to the most up-to-date algorithms.

The structured and modularized nature of MUVES code has reduced the historical difficulties associated with code maintenance. Software modules with well-defined interfaces have been written to perform singular tasks. Modules have been combined into software packages; each package contains software related to a similar set of tasks. When modifications are required, only changes to the individual module or affected packages will need to be made. Software modules reduce the potential for comparatively large interacting sections of code that would be more difficult to maintain. A standardized MUVES interface will allow code developers to insert special components, experimental materials, and new models into the system with minimal changes to existing code. This is one way that MUVES improves maintainability over the plethora of existing vulnerability analysis codes.

MUVES also provides an environment in which a vulnerability analyst can explore new situations, new methods of analysis, and experimental applications within a single computational framework. It should be possible to handle special requirements for vulnerability/lethality analyses without having to modify *any* software. The code is general enough so that in most situations an analyst only needs to modify input parameters or request different options via the user interface to obtain the desired results. Unusual situations, however, may require code extensions to MUVES. For example, a new concept of target-threat interactions may require new modules. In all phases of design and coding, provisions have been made for future extensions of the MUVES code.

Compared to previous computer models, modifications to MUVES will be more tightly controlled. There will be an "official" supported version of MUVES both in source and binary forms. In order to maintain consistent results throughout the VLD, the installed version at the BRL will be identical across computer systems and will be used for all routine production runs. Analyst contributed code may eventually be integrated into the supported version of MUVES after being reviewed for compliance with MUVES system interface requirements.

The Source Code Control System (SCCS), a collection of utilities that runs under the UNIX operating system, is being used to control and account for all changes to MUVES software and documentation. Each release of MUVES will have a version identifier that can be used to reconstruct the set of sources used to generate the release.

Over the past decade, the VLD has invested in-house resources to develop an interactive geometry editor based on Combinatorial Solid Geometry techniques [9]. This editor, the Multi-device Graphics EDitor (MGED), has become a standard BRL tool for describing targets for vulnerability/lethality analyses. The BRL has also developed a ray-tracing package to interrogate objects in an MGED database. One application of this package permits renderings of MGED objects on a graphics display, *e.g.*, color shaded images of target views [10]. Combined, these two packages are useful for describing and viewing target geometry inputs for vulnerability/lethality studies. MUVES exploits these facilities as well as the existing library of target descriptions. Recognizing that other geometric modeling techniques and ray-tracing packages may need to be supported, MUVES is not intrinsically tied to these facilities.

Through the MUVES user interface, the analyst works with a hierarchy of menus to accomplish such tasks as selecting input parameters, specifying analysis methods, and identifying desired outputs. Whenever possible, default values are made available for selection. An analyst can edit input values and save them for use in later runs. In addition to traditional keyboard methods of entry, a graphical user interface may be eventually developed for interacting with MUVES.

Care has been taken to ensure that MUVES facilities can be accessed from a wide variety of computer terminals and interactive workstations. The first release of MUVES provides a menu-driven user interface which can be displayed by a wide variety of ASCII terminals.

The overall result is that MUVES provides an integrated framework that can grow in a controlled and maintainable way as the vulnerability/lethality community uncovers new situations, develops new methods of analysis, and explores experimental applications.

1.3 Project Description

In order for MUVES to satisfy the identified project goals, software engineering principles concentrating on the flow of data and the transformations on this data were used extensively. A consistent philosophy towards the problem of assessing the remaining utility, *i.e.*, functionality, of a target damaged by some

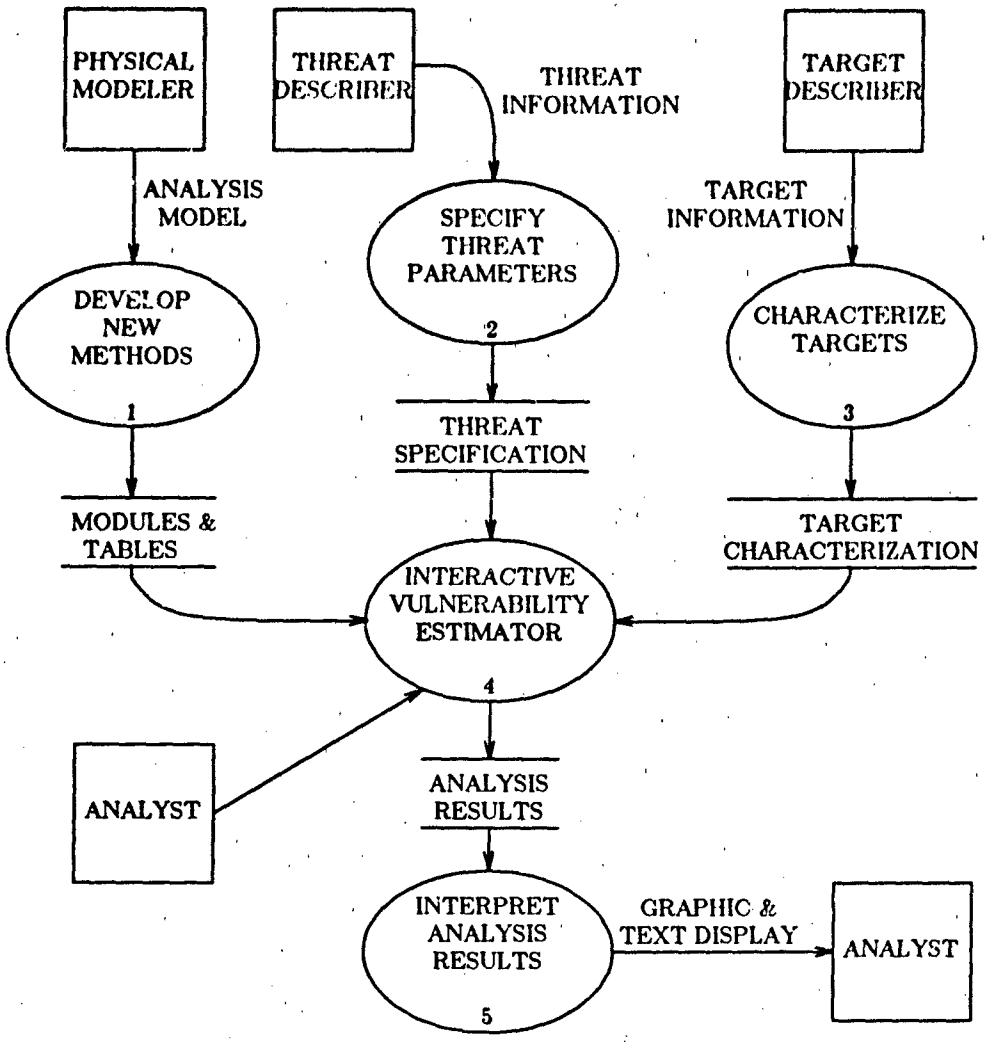


Figure 2. Top-level MUVES description

sort of threat was employed. This approach has helped clarify many of the conceptual difficulties which have plagued previous vulnerability analysis efforts. The first aspect of this philosophy was to generalize the statement of the problem in such a way that most vulnerability problems could fit into a single, generic framework. This framework is stated as follows:

> "An object, called the threat, is thrown at another object, called the target, in such a way that its trajectory can be approximated by a ray; some sort of damage occurs to one or both objects as a result of the interaction between the two objects, and possible new threats are created; then the utility of the damaged target is assessed."

At the top-most level, this process can be depicted as in Figure 2. Note that this entire process can be described by the flow of MUVES inputs, the processing that MUVES performs, and the outputs that it produces.

In this figure, one or more individuals may share the responsibilities required for an analysis. These responsibilities include: (1) the vulnerability experts (physical modelers) who are responsible for developing and designing new methods, (2) the threat describers who test and evaluate threat information, (3) the target describers who assimilate target information into a complete characterization, and, finally, (4) the analysts who select input parameters, the methods of analysis, and the desired outputs. Regardless of the type of model, information from *each* of these four input sources is necessary for *any* interactive vulnerability estimator. In Figure 2, inputs which are stored on-line are partially enclosed by two parallel lines. To complete an analysis, results are displayed and interpreted by vulnerability analysts.

One of the major MUVES concerns is characterizations of threats and targets, and the events occurring at the moment of impact of a threat with a target. To make the interaction aspect of the problem easier to change as new data becomes available, consideration of *all* threat-target physical interactions were isolated into one area within the MUVES framework. In MUVES terminology, this area is referred to as the "interaction module". It is important to realize that this is not one module but a large number of modules, each of which performs the calculations for a specific type of threat interacting with a specific target component.

The components of interest and the manner of evaluating damage to them varies with different types of analyses. Therefore, there will be a different family of interaction modules called upon to do the calculations for each type of analysis. Many of these internal modules are likely to belong to more than one such family. This is especially true for frequently used modules such as those for armor perforation calculations. Thus, MUVES will eliminate redundancies evident in earlier vulnerability programs and will provide a previously unavailable flexibility.

The philosophy used to design MUVES, however, places certain restrictions on what can be done within the MUVES. For instance, MUVES is currently not able to model all effects since all ray-tracing within the MUVES environment is currently limited to straight rays. Modeling events such as diffusion of gases and temperature radiation is not feasible since these events consist of fronts of molecules or energy moving through varying media. Similar difficulties arise with blast and shock effects. These may be approximated using large numbers of rays radiating from a single point. Although this technique is computationally intensive, it could be used to study these phenomena for complicated target geometries until better methods can be devised.

From this preliminary design step, the structured analysis phase of this project focused on defining the needs of the users of the eventual product. An attempt was made to produce a generalized description covering everything that must happen when analyzing the vulnerability of any target versus any direct impacting threat. Previous efforts to revise vulnerability methodology have frequently missed this generality by concentrating on hardware limitations or specific assessment techniques too early in the design phase. The result of the structured analysis phase was a generalized description which should meet all current analytical requirements with few changes in the necessary inputs. This description also eliminates most of the redundancies and streamlines the entire vulnerability/lethality process.

The products of this phase of the project included detailed data flow diagrams (such as the data flow diagram shown in Figure 2), and explicit definitions for the data flows and the processes. These definitions specify the exact composition of the data, the internal logic of these processes, and the specific function(s) of every process. From this phase, the MUVES coding phase began. Readers who are further interested in the design products should consult the listed reference for additional information [6].

1.4 Advantages

MUVES provides several advantages over previous vulnerability/lethality codes. These advantages include: (1) a single framework for vulnerability/lethality models, (2) a user-friendly interface which organizes all data, (3) a high degree of analyst control over processing choices, (4) the ability to select shot patterns, (5) on-demand analysis of subsidiary threats, (6) the isolation of threat-component interactions, (7) the isolation of engineering approximation techniques, and (8) opportunities for more complex forms of damage assessment.

Within the MUVES framework, rays representing shotlines are used to simulate threats intersecting target components and to assess damage at those intersections. In MUVES, a component is *any* identifiable piece of the target having definite geometry which should be analyzed as a *unit* with respect to its interaction with the threat. Thus, for a compartment-level analysis, a compartment may be considered as one rather large component, though it may consist of many individually modeled objects. For point-burst analyses, the individual components may still be treated separately.

This way of describing components gives the target describer the flexibility to build large components for a coarse analysis from several smaller components which are typically useful for more detailed analytic methods. This permits the analyst to generalize certain aspects of the analysis while retaining detail only where it is necessary. It also allows an analyst to use a single target description for several methods of analysis and change only the listing or grouping of objects in the MUVES input files to achieve the desired result. The BRL Multi-device Graphics Editor also offers some support for maintaining a collection of geometric descriptions with varying degrees of resolution for a single target [9]. However, there is a limit to how far this approach can be pushed. For instance, some analysis methods may require geometric detail that would unduly slow computations for other approximation methods.

One of the most readily apparent improvements of MUVES over the current methodologies is that the analyst can choose from a variety of available shot patterns for the threat, rather than being constrained

to a single pre-determined pattern (*e g.*, a simple grid). There are four different types of shot patterns currently available: (1) a rectangular grid with flexible parameters (including perspective rays), (2) randomly distributed patterns (*e g.*, a bivariate Gaussian) around a single aim point, (3) spherically distributed start points, and (4) an analyst specified set of shot points. Analysts can select a shot pattern appropriate to the problem, rather than trying to fit the problem into a more limited set of alternatives.

There is a feedback loop within the MUVES analysis process which returns newly generated threats (*e g.*, spall, deflected main penetrator, *etc.*) to the ray-tracing sub-process when they are created by the threat-target physical interaction module. Prior computer codes calculated all shotlines and trajectories of potential spall rays before any analysis was performed. The MUVES approach eliminates unnecessary computations (*e g.*, tracing spall rays when the main penetrator fails to perforate the armor).

Each threat-component interaction is handled by the appropriate interaction module. The interaction module computes the effects of a specific threat impacting a component. Within an interaction module, threat parameters may be altered, damage may be produced for the component, and new threats may be generated. Analysts have the ability to select module preferences via the user interface when multiple modules exist for a particular threat-component combination.

When all damage producing interactions have ceased, the evaluation phase begins. The modules that estimate the functionality of damaged target components are termed evaluation modules. Analogous to interaction modules, these modules also have interfaces which are well-defined and standardized. Damage is typically evaluated based on engineering techniques. Input files and selections from the user interface control the damage assessment process, and in most cases, damage assessment requirements do not require modifications to the underlying code. Compared to earlier codes, analysts may develop new expressions in the input files for assessing damage and can immediately test these changes by selecting the new files and running the analysis. The structure of the input files and the underlying software modules also provide analysts with the opportunity for more complex forms of damage assessment.

1.5 Configuration Management

In the mid 1980's, the BRL's VLD recognized the need to standardize, reorganize, and rigorously manage the configuration of the compartment and point-burst models to ensure that vulnerability analysts using these codes would be producing consistent and auditable results [11]. While MUVES is designed to facilitate solutions for these requirements, these requirements are not completely met by the initial release of MUVES. Most notably, the first release of MUVES does not provide the point-burst vulnerability models. Although the underlying code contains an interface for archiving/retrieving data and analysis results, the interface requires additional testing before being released. The former will be addressed by the MUVES development team in a subsequent release; whereas, the latter is currently being addressed by a database team (with MUVES interface specifications being provided by the MUVES development team).

MUVES requires target files, threat files, damage evaluation curves, and other input files be imported into the MUVES file hierarchy. Project directories are used to store related inputs and outputs. MUVES provides automatic mechanisms for storing input files, session files, and results files in this hierarchy.

Session files list the inputs used in the process of creating the corresponding results files.

The first release of MUVES allows analysts to copy files into and out of the MUVES hierarchy. The ability to copy files from the MUVES hierarchy gives analysts the ability to edit analysis input files and to copy new versions of these files into MUVES for use in subsequent analyses. Given vulnerability analysts typically perform numerous analyses before settling on one or more runs containing the final information to be delivered to the customer, MUVES allows analysts to permanently remove any input, result, or session files which are no longer necessary. Although MUVES caches results and session files in its own directories, analysts may copy or remove these files. A future release of MUVES will provide an interface for additional access and archiving control.

In summary, it cannot be overemphasized that the first release of MUVES provides the minimum acceptable level of audit controls and that most of the responsibility for this process lies with vulnerability analysts. MUVES developers, on the other hand, have the responsibility to standardize, reorganize, and rigorously manage the configuration of the vulnerability/lethality models to ensure that the MUVES code is producing consistent and correct results. Needs have been identified for the establishment of formalized procedures to insure work is safely stored and to insure the final system is responsive to the needs of the production work force.

1.6 State of Development

The initial release of MUVES does not incorporate aspects of the point-burst and the stochastic point-burst models. Rather than delaying the introduction of MUVES until these efforts are completed, a decision was made to release MUVES at the earliest possible date where it would be an *effective* tool for performing certain vulnerability/lethality studies. The first release of the code was defined as the completion of the compartment model for target-threat interactions [6].

Future development efforts will exploit the MUVES framework. A list of the critical and desired capabilities for the next release of MUVES has been created. This prioritization scheme emphasizes incorporation of the stochastic point-burst model and additional compartment model enhancements for different armor packages and threat types. As with all projects, changing requirements necessitate periodic review and restructuring of priorities. Vulnerability analysts are encouraged to continually participate in this process and to supply their insights on the available MUVES capabilities, "must-have" requirements, and desired capabilities.

1.7 Additional Documentation and Training

To adequately describe MUVES for vulnerability analysts, system administrators, and code developers, the roles of these individuals will be discussed in a separate volume of MUVES documentation.

Vulnerability analysts who feel that this guide is incomplete in certain areas or who have never had the opportunity to read some of the references listed in each chapter/appendix of this guide are encouraged to

locate copies of these references and to explore the information that they provide.

Training classes for using MUVES will be conducted from time to time for BRL employees and other interested personnel. Interested parties should contact the MUVES Development Team to discuss training requirements and schedules.

1.8 Software Distribution

Inquiries should be directed to the BRL/VLD Vulnerability Methodology Branch at mdt@brl.army.mil or Director, Ballistic Research Laboratory, ATTN: SLCBR-VL-V (MUVES Development Team), APG, MD 21005-5066.

2. System Design

This section describes the overall design of MUVES; it gives a "bird's-eye view" of the system. The general concepts guiding the structure and operation are discussed in the first three subsections. There is also a subsection covering the outputs available from MUVES.

Another section of this guide that is helpful for gaining an overall understanding of MUVES is the **Glossary**, which contains special terms used in MUVES documentation. Any terms found in the glossary are italicized the first time they appear. A working familiarity with these terms is a prerequisite for discussing MUVES. This section should provide enough contextual information to make the glossary understandable.

2.1 Structure

There is a large quantity of data flowing through the MUVES analysis process. However, the general structure of the program can be expressed in the following Data Flow Diagram:

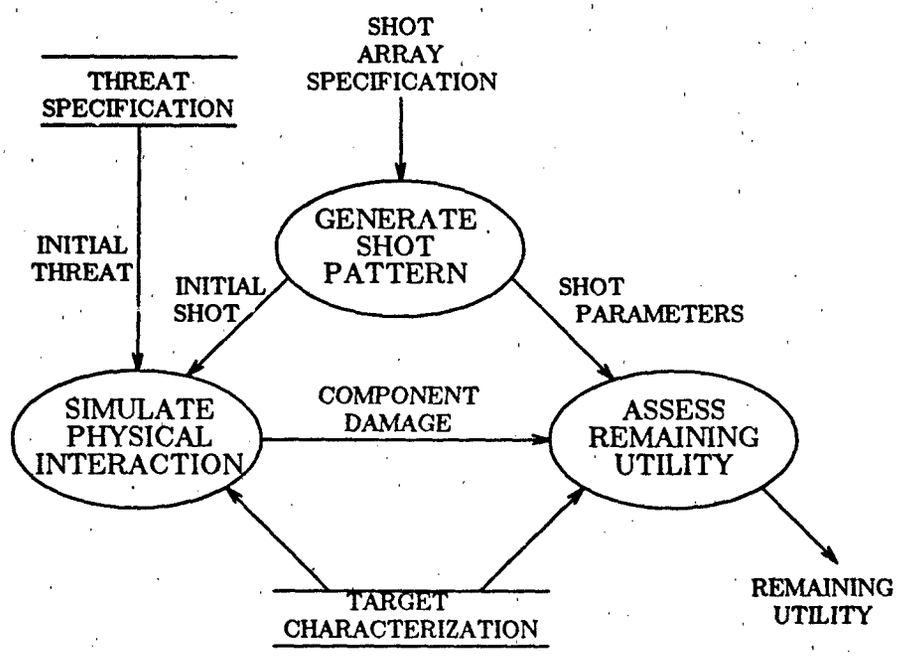


Figure 3. MUVES Analysis Principal Data Flows

In the most general terms, MUVES sets up the initial geometric and physical parameters of a threat, determines the results of an interaction between the threat and the target, and evaluates damage to the target based on that interaction.

The threat/target interaction is computed using a *threat path*. A threat path consists of geometric information for the components along a (not necessarily straight) path through the target plus a set of parameters describing the physical attributes of the threat in question. The geometric information for a single component is called a *component trace*; a trace is a straight line through a component, containing the coordinates and normals at each intersection. Although each trace through a single component is linear, the threat path may be constructed from non-collinear traces through different components. The parameters describing a particular threat are stored in a data structure called a *threat packet*. These structures are keyed to specific types of threats (*e.g.*, kinetic-energy or shaped-charge) and sometimes to the algorithms used for evaluating their effects (*e.g.*, Fireman-Pugh versus DSM evaluation for shaped-charge munitions); other information, such as geometric values, may also be stored in these packets as needed.

The initial ray tracing is determined by the *view* specified by the analyst. A view specification may consist of a number of individually specified shots, a pattern of shots (*e.g.*, a grid) relative to some defining orientation, or some combination of the two. After each ray in the shot pattern is traced through the target, the initial parameters of the threat are attached to the first component on the path, and this collection of data (now called an *initial threat path*) is passed to the interaction subroutines.

MUVES traverses the threat path computing interactions of the threat with each component in the path as it is encountered. At each component, several things may occur: (1) damage may be recorded for the component, (2) the threat path may be altered, usually by modifying the threat parameters and propagating them to the following component, and (3) secondary threats may be created. If the interaction indicates that a threat must change direction, this may be done by ray tracing the new direction and attaching the threat parameters to the set of component traces produced along the new ray trace. This process is repeated until the threat no longer retains any damage-producing capability or there are no more components in the threat path. Along the way, secondary threats may be generated and their paths ray traced to produce new threat paths; these are processed in the same fashion until all damage-producing mechanisms have terminated. Damage to components caused by any threat is recorded and stored until all interactions for the shot have been processed. This damage information is recorded in data structures called *damage packets*, which define the type of resultant damage (*e.g.*, blast, main penetrator impact, or spall fragments) and contain any parameters needed to describe the degree of damage. Damage is recorded only for those components which are deemed "critical" for the *mission function(s)* specified by the analyst. A *critical component* is any component whose loss affects the target's ability to perform its designated function. In order to be considered critical, a component must be referred to in an assessment expression (described in the next paragraph) used in the current analysis session.

Finally, some measure of the target's ability to function is evaluated from the damage to the individual components. This is referred to as the target's *functionality*. The damage for each component is collated, and a utility value is computed from the damage to that component during the interaction phase. Systems within the target are defined in *system definitions*, which are Boolean or mathematical combinations of component and subsystem utilities. The utilities of the systems in the target are evaluated based upon the utilities of their constituent components. The functionalities of the target are then computed from the relevant system utilities. These functionalities are determined for one or more *contexts*, which is the term

used for the combination of mission function the target is expected to perform and the environment in which it must operate. The contribution of each system to the overall functionality of the target in a given context is defined in a *damage evaluation expression*, which is also a combination of component or system utilities.

The syntax for the system definitions and damage assessment expressions are similar. The mini-language (defined in Section 3, **Analyst Supplied Input Files**) used for these expressions provides the flexibility to specify existing assessment combinations while permitting forms of evaluation that were not previously possible without editing and recompiling source code.

2.2 Approximation Methods

Each MUVES analysis is conducted within a category of computations called an *approximation method*. An approximation method is the collection of assumptions, simplifications, empirical data, and mathematical models used to approximate the physical processes involved in the interaction of the threat with the target. All vulnerability analyses fall into some class of approximations (compartment, point-burst, stochastic, *etc.*). Traditionally, these have been referred to as "models" or "methods" and implemented as separate programs. The MUVES developers coined the phrase "approximation method" to refer to these various packaged methods of analysis.

In MUVES, approximation methods are partly implemented as a set of modules included within the main program which are invoked for specific combinations of threat and component. When determining threat-component interaction results along a threat path, MUVES selects from the available set of modules to obtain the algorithm to use for computing the results of the interaction. The selection is based on the *threat type* and *component category* for each trace in the threat path. Threat type is a grouping used for threats which behave in the same fashion when interacting with components (*e.g.* shaped-charge jet). The modules used for this class of computations are called *interaction modules*. Interaction modules propagate a threat through the target and determine what happens at each juncture; they determine the physical parameters (*e.g.*, hole size, number of fragment impacts) of the damage done to each component by these interactions.

The modules for computing utility values for damaged components are called *evaluation modules*. They are specified by means of the *damage evaluation selection* file. The damage evaluation selection file contains the names of every *component category* in the target and the evaluation module to be used to compute the utilities for components in each category. A component category is, by definition, a grouping of components for which damage is computed in an identical fashion. These must be chosen from the subset of evaluation modules available for the particular approximation method, but the damage evaluation selection file gives the analyst significant latitude in tailoring the analysis for specific purposes.

The data in threat files and most target input files are keyed to specific approximation methods. In the general case, the threats appropriate to a given method may differ significantly from those used for other methods (*e.g.*, kinetic-energy projectiles to a compartment or point-burst method, neutrons to a neutron transport model, electromagnetic waves to a radar model), so care should be taken when selecting threat

files for a project. In order to maintain consistency, each project should use only one approximation method, unless making explicit comparisons between approximation methods. This should help prevent confusion when examining archived results. There is a **samples** project where examples of input files for the compartment method may be obtained; this project is available through the user interface.

2.3 Processes

There are a variety of processes running during a MUVES analysis session. The most visible, and usually the only process the analyst deals with directly, is the user interface. The user interface provides a convenient method for specifying the parameters for an analysis. The analyst is not required to know the interface specifications used by the MUVES analysis process; making it possible to concentrate on the analysis task rather than grammar. The analysis information is stored in a *session file* so that it may be re-used at a later date to initiate further MUVES runs. The user interface allows the analyst to specify any previous session file for use in the current MUVES run (he may also annotate the session file for future reference).

The user interface also performs the configuration management functions required in the MUVES charter. It maintains the correlations between the session files and their corresponding results files, plus any required input files. In the future, when the analyst wishes to archive results, the user interface will search the appropriate session files and archive all of the relevant inputs along with the final results. If files used in an analysis have not been archived, the user interface will print a warning message on removal attempts and will not remove anything without further confirmation. The goal of this behavior is to guarantee the ability to reconstruct an analysis at some future date and to help prevent accidental loss of useful information.

The MUVES user interface exploits the UNIX file system to organize file storage for all its requisite files. All MUVES-related files are located within a single tree rooted at a directory chosen when MUVES is installed. For proper operation of MUVES software, users must set a **MUVES** environment variable equal to the absolute pathname of this root directory.* In the remainder of this document, the value of the **MUVES** environment variable will be indicated by **$MUVES**, which is the way it would be used when typing shell commands. In addition, users must include **$MUVES/bin** in their **PATH** environment variable.† For additional information regarding environment set up, MUVES users should consult the "MUVES System Administration Guide" or their local MUVES system administrator.

---

* For example, if the installation root directory is `/usr/muves`, then UNIX Bourne shell users would mostly likely place the following command in their .profile automatic login shell configuration scripts: `MUVES=/usr/muves export MUVES` followed by `PATH=$MUVES/bin:$PATH export PATH`.

† Similarly, if the installation root directory is `/usr/muves`, then users would include `/usr/muves/bin` in the `PATH` in their login shell configuration scripts.

Note that maintaining any degree of configuration control over the analysis results requires that the analysis be conducted through the user interface. This may seem onerous at first, but the long term benefit of this kind of record-keeping is one of the primary reasons for the initiation of the MUVES effort.

The heart of MUVES is the analysis program, which is called "muverat" (for historical reasons). This program is the portion of MUVES that manages the threat-component interactions, computes component damage, and evaluates the remaining target functionality for each context desired. It then writes these functionalities to the specified results file, the name of which is determined by the user interface.

The MUVES analysis process initiates a separate process to ray trace the target geometry; the two processes communicate via UNIX pipes. This approach to ray tracing allows MUVES to use network channels to exploit resources beyond those on the computer running the user interface and main analysis code. In this way, ray tracing, the most computationally intensive portion of the analysis, may be performed on larger machines while keeping the inputs and results in a standard working environment. Thus, for a large target, the analyst could specify that ray tracing be done on a supercomputer, while the array of shots is analyzed on a desktop workstation. MUVES is designed to perform this type of networking with minimal effort on the analyst's part.

It is necessary to sort the component damage for each shot in order to insure proper evaluation of damage to each component (all of the damage for a particular component must be grouped together). But, because the quantity of damage produced for the components can be quite voluminous, a separate process may at times be created to sort the damage resulting from the interactions. This process is called "cdsort" and exists for no other purpose than to sort component damage for "muverat"; it is only invoked when memory limitations on the current machine make it necessary.

2.4 MUVES Output Files

Ultimately, all of this computation produces a series of values describing the target's vulnerability when attacked by the threat from various attack aspects. These values are called the *final results* and are stored in final results files; these files contain only the vulnerability information for the target. Another type of output, called *intermediate results*, provides insight into the progress of the computations through the various interaction and evaluation modules.

2.4.1 Final Results Files

The final results file contains the functionality results for each shot against the target. They are the primary results obtained from MUVES. Many other output formats may be derived from the results stored in this file.

The final results file contains a header describing the contents of the file, followed by a series of shot records divided into views. The header allows various programs using these files to index into the contents of each file. Each shot record contains the origin point and direction defining the initial ray trace for each shot that hit the target, followed by the functionality results for each requested context. A view is a

convenient divider delineating a collection of shots derived from a single shot group evaluating the effects of one threat against one target, using one approximation method.

The functionality values are given in the form of fractional remaining functionalities or FRFs. FRFs are the complements of losses of function (LoFs) used in historical vulnerability analysis programs. The exact meaning of these numbers is left to the approximation method designer, but in general they define the ability of the target to function after it has been hit by the threat in question. The primary reason for using the complement of the traditional values is that it allows the analyst to write his system definitions and damage assessment expressions in terms of positive combinations rather than negative; this makes the expressions easier to write and thus less error prone. For certain methods, however, the interpretation of these values may differ significantly; one should always carefully read the documentation for a method before using it.

The final results file is intended to be a generic storage file for the basic vulnerability information obtained from a MUVES analysis run. Although these files are readable, they are not particularly easy for a person to digest. Therefore, there are a number of postprocessors available to make these results more useful to the analysts; these are described in Section 5, **Postprocessors**.

2.4.2 Intermediate Results Files

The final results files provide the end product of the computations, but they omit internal details of the computations that are occasionally useful to the analyst. Such internal information may be useful when debugging a target description, developing a new methodology, or simply investigating the causes of unexpected results. Since one does not want such information in a typical analysis, there is an option in MUVES (which may be selected from the user interface) to write out intermediate results.

When the intermediate results option is on, the analysis process records detailed information regarding the geometry, threat parameters, interaction, and component damage for each trace of the analysis. To avoid performance penalties, this information is written in binary format; therefore, an interpreter is required to read the information in these files. The program to do this is called `ir2ascii` and is described in the Section 5, **Postprocessors**.

When recorded for full-view analyses, these files can become quite large (tens of megabytes). Therefore, analysts are encouraged to use this option only when necessary and to remove the files as soon as they are finished with them. Also, it is suggested that analysts select a small set of relevant shots when recording intermediate results; this will make it easier to interpret the resulting file.

3. Analyst Supplied Input Files

Before the development of MUVES, target and threat inputs for vulnerability/lethality studies were maintained by a handful of individuals. To insure data integrity for vulnerability/lethality studies with MUVES, it is still envisioned that only a few individuals will be responsible for entering and maintaining target and threat files. For completeness, all files that an analyst must specify for a MUVES analysis are described in this section.

To perform an analysis with MUVES, analysts must supply the following types of files: (1) the initial threat parameters describing the characteristics of the phenomena capable of damaging a target, (2) the target parameters and ray-tracing inputs, (3) the interaction and evaluation curves for modeling and evaluating the target-threat interaction. MUVES uses information from each of these files to simulate the physical target-threat interaction and to estimate the functionality of damaged target components. This section presents the general file formats. Details specific to a vulnerability assessment methodology, *i.e.*, approximation method in MUVES terminology, will be discussed with each method.

The MUVES file formats in this section will *always* be explained by carefully designed examples. In most instances, these examples will be supplemented with Backus-Naur Form (BNF) grammars that concisely describe the file syntax [12]. Familiarity with BNF notation is not required to understand the file formats as they are presented in this guide. Analysts may skip the sections where the file formats are summarized in this notation ***without any adverse effects***. However, BNF grammars are provided since they are a convenient short-hand notation, *i.e.*, they conveniently depict tabs, white spaces, and repetitive sequences.

Analysts who uncover inconsistencies or errors with *any* centrally maintained files (particularly, threat files and target geometry databases) are responsible for contacting the individual(s) who maintain these files. To assist with the iterative process of identifying and correcting inconsistencies, MUVES has been designed to allow analysts to manipulate personal copies of *all* input files and to apply patches to their copies until the central files can be updated. To successfully manipulate copies of these files, analysts must understand the purpose and syntax of each file.

3.1 Input File Syntax

The syntax for every MUVES input file is based on a few simple rules. *First*, all filenames enclosed in double quotes in this documentation correspond to the actual filenames that MUVES searches for in its file hierarchy. Other input files in the file hierarchy are not required to have specific names. Also, certain input files may not be required for a given analysis. The MUVES user interface insures that new files are entered in the appropriate location within the file hierarchy. When naming files, analysts should not use filenames beginning with a '.' (*i.e.*, a period) and should not exceed the maximum length for a filename

supported on their particular system.* *Second*, all alphabetic and alpha-numeric strings in any MUVES input file are (upper and lower) case sensitive. *Third*, blank lines and comment lines, *i.e.*, lines starting with a '#' character, are acceptable in any input file. The remainder of any line which contains a '#' character immediately preceded with a tab character is also a comment. By definition, the remainder of any line which contains a '#' character immediately preceded by any other character (*e.g.*, a space) is *not* a comment. All lines consisting of only a comment are automatically skipped over by the MUVES software during processing. *Fourth*, the MUVES software frequently needs to look-up values from tables or interpolate a value based on table entries. For consistency, MUVES requires all files containing interpolation tables to adhere to the remaining rules in this section.

All interpolation tables determine the value of a dependent variable (*i.e.*, Z) based on one or two independent variables (*i.e.*, X or X,Y). Each interpolation table should be tagged with a distinct one-line label, *i.e.*, a table tag. Table tags are the only means for locating an interpolation table in a file which contains more than one interpolation table. A table tag is optional; however, if a table tag is not specified or there are no tags in a file containing one or more interpolation tables, the first table in the file is always used. If identical table tags are mistakenly used for several tables in a file, only the first table encountered in the file with that table tag will be used.

Table tags may not start with a '#' or a '!'. Leading space or tab characters on a table tag line are ignored.

Each non-tag line in a table file consists of non-empty fields separated by spaces and/or tabs. Following the optional tag line, the first non-empty/non-comment line in the table must contain the dimension(s) of the table. This line must have a '!' in the first field followed by either one or two numbers, depending on whether the table is one or two-dimensional. The number of dimensions specified for the table must correspond to the number of table entries in the file. Missing entries in the table are not permitted. The table dimension(s) may be followed by one or more flags which control the method of interpolation for dependent values not specified in the table.

Table data values provide a set of known values for a function of one or two variables. For points other than those at which the known values were provided, an interpolated value is computed using nearby known values. Optional flag(s) may be used to control the method of interpolation for these points.

If flags are specified on the dimension line for a table, then those flags define the method of interpolation for the table; otherwise, the default flags for the approximation method will be used. To ensure all data is interpolated as expected, it is highly recommended that flags be specified for all interpolation tables. Table-file flags and their meanings are as follows:

---

* If the maximum filename length is not known for a particular system, an upper limit of 14 characters for a UNIX filename guarantees uniqueness. Recent versions of the UNIX OS typically guarantee uniqueness for quite many more characters.

TABLE 1. Interpolation Table Flags

okay	**issue no warning message when table domain is exceeded. It is acceptable to exceed the domain of the data.**
warn	**issue warning message if table domain is exceeded.**
step	**interpret the table as a step function**
extrapolate	**extrapolate in all directions**
limit	**limit to table edges in all directions**
xfirst	**limit to the first X margin value**
xlast	**limit to the last X margin value**
yfirst	**limit to the first Y margin value**
ylast	**limit to the last Y margin value**

When the **okay** flag is specified, warning messages will not be printed. The **warn** flag should be specified if a warning message is desired when an evaluation lies outside the bounds of the table. Only the first violation of each limit on a table will be reported.

The **step** flag indicates that the data in the table describes a step function. Each specified Z value applies from the corresponding margin value (inclusive) up to the subsequent margin value (exclusive) in the order specified in the table data file. For a two-dimensional step function, this rule applies to both margins. Edge-limiting requests (including **limit**, **xfirst**, **xlast**, **yfirst**, and **ylast**) are not permitted for step functions. Out-of-domain evaluation of a step function is always mapped to the nearest valid interval anyway. If **step** is not specified, linear or bilinear interpolation using the nearest known values may be performed.

Extrapolation consists of using the data nearest the appropriate edge to evaluate the function for points beyond the bounds of the margins. When the individual edge-limiting flags are specified, extrapolation is performed in all other directions. The **limit** flag disables extrapolation in all directions. The **yfirst** and **ylast** flags are not permitted for one-dimensional tables.

Suppose **extrapolate** and **warn** flags are the defaults for all tables for a particular approximation method. In this case, specifying only the **okay** flag for a table will cause extrapolation in all directions without printing any warning messages.

A one-dimensional table must have two columns of equal length. The independent variable is in the first column (the X margin) and the dependent variable is in the second column (the Z(X) values). Figure 4 contains a one-dimensional table with four rows. The independent variable X is given in strictly

```
# Example: 1D table file

! 4     limit okay

# X:     Z(X):
120      .3
131      .4
143      .7
157      .9
```

Figure 4. A sample one-dimensional table file

ascending order and is separated from the dependent variable Z(X) by spaces/tabs. The table would also be acceptable if the independent variable X was given in strictly descending order. The **limit** and **okay** flags indicate that interpolation is to be limited to the table edges in all directions (*i.e.*, for all X values greater than or equal to 157, the interpolated value will be .9) and that no warning message should be displayed if the table domain is exceeded.

```
# Example: 2D table file

! 4 3   extrapolate     warn

# Y margin values:
          12                15                18
# X:    Z(X,12):          Z(X,15):          Z(X,18):
 120      .3                .35               .4
 130      .4                .4                .3
 140      .7                .6                .5
 150      .9                1.85              2.8
```

Figure 5. A sample two-dimensional table file

A two-dimensional table consists of a row of independent variable values (the Y margin), a column of independent variable values (the X margin), and a corresponding table of dependent values (the Z(X,Y) values) for each combination of X and Y. Figure 5 contains a two-dimensional table with four X values and three Y values. Notice that both the X and Y values may be preceded by any number of spaces or tabs.

A table file with more than one table typically consists of a sequence of table tags, each followed by a one-dimensional or a two-dimensional table in the preceding formats (Figure 6). Margin values for the independent variables (X,Y values) are not required to be spaced at regular intervals. However, margin values must be given in either strictly ascending or descending order. In addition, spaces and/or tabs must separate independent and dependent values. Acceptable table tags are discussed with each approximation method.

```
# First table
#   (Note: Leading space is not part of the tag.)
       axle
! 4 3   limit   warn    # limit in all directions
        12      15      18
120     .3      .35     .4
130     .4      .4      .3
140     .7      .6      .5
150     .9      .85     .8

# Second table
#   (Note: The spaces and # is part of the tag.)
side plate # 3
! 3     xfirst  warn    # only limit "xfirst"
140     1.5
130     2.6
120     3.5

# Third table
#   (Note: The tag line is identical to the tag of the second table.
#          Tags should be unique within a file but if they
#          are not, the first table in the table file with
#          a given tag will be used.
       side plate # 3
! 1     warn
130     2.5
```

Figure 6. A sample table file containing multiple tables

3.2 Threat Information

To describe the pertinent characteristics of a threat engaging a target (*e.g.*, kinetic-energy projectiles, shaped-charge munitions, explosively formed penetrators, fly-over and shoot-down munitions, *etc.*), MUVES utilizes threat parameter files. For each threat, analysts must specify the type of threat (*e.g.*, kinetic-energy) and construct one or more threat packet(s) which provide the initial parameter values for the underlying computational algorithms. For each threat, the threat packet file must be named "initial". For the several classes of threats already incorporated into MUVES, supplemental interpolation tables relating one or more parameters are also required to describe the threat. All files which are required to completely describe a particular threat, including the "initial" file, are stored in a threat directory. Thus, specifying a threat for an analysis equates to selecting this directory and its files.

Regardless of the type of threat, MUVES expects *all* threat parameters to be in *metric* units (as opposed to English units). For each threat parameter, specific metric units are assumed. Violation of the metric unit assumption or the specific units for a particular parameter will produce meaningless analysis results. The underlying software is typically incapable of detecting these types of errors.

It is also important to understand that threat files do not select the algorithm preference(s) for modeling the target-threat interaction. In cases where different algorithms may be used, MUVES will provide the analyst with the opportunity to select the computational algorithm. Provided the algorithm-specific packets exist in the "initial" file, MUVES will automatically use the appropriate threat packets for

the user-selected computational algorithm. If an algorithm is not selected, a default algorithm will be used. The default algorithm preference(s) are discussed with each approximation method. If the packets for the default algorithm do not exist, appropriate error messages will be issued.

As new vulnerability assessment methodologies are incorporated into MUVES, the ability to model and study new threat types will be added. For completeness, the types of threats and their associated input files will be discussed with each vulnerability assessment methodology. However, to provide analysts with an understanding of the basic type of information contained in threat files, examples of "initial" and supplemental files, such as the ones used for Compartment-type analyses, will be presented in this section.

3.2.1 Initial File

An "initial" file specifies the threat type and provides one or more threat packets which are required for the type of analysis specified. To understand the structure of this file, it is first important to note that lines beginning with a '#' character and the remainder of lines containing a tab followed by a '#' character are comments and are ignored.

```
threat_type      # Name for the threat type

#       Threat packet and its parameters

packet_name      # Threat packet name
 string          # e.g., threat name
 boolean         # i.e., true or false, 1 or 0, t or f, y or n, on or off
 float           # e.g., 5.21e4
 float           # e.g., -2.1
integer          # e.g., 1
endpacket
```

Figure 7. Generic "initial" file

Within the "initial" file (Figure 7), MUVES requires each item to be specified on a single line. The first entry in an "initial" file must specify the threat type. Acceptable threat types for a particular analysis will be discussed with each approximation method. Following the specification of the threat type, MUVES expects one or more threat packets. The names and number of the threat packets for a threat type is dependent on the approximation method and the number of computational algorithms which can model the target-threat interaction.

For instance, there are presently two computational algorithms in the MUVES "compart" approximation method to model the penetration of a shaped-charge jet (SCJ). These algorithms are the Fireman-Pugh algorithm [13] and the DiPersio, Simon, Merendino (DSM) algorithm [14]. Thus, two separate SC threat packets specify parameters for the Fireman-Pugh and DSM algorithms.†

† When a SC warhead is selected through the MUVES user interface, analysts may select the algorithm (and corresponding threat packets) to model the damage if the SC warhead fuzes.

Every threat packet begins with a threat packet name and terminates with **endpacket**. The parameters in each threat packet are expected to be in a specific order; however, the ordering of the threat packets in the file is not important. Threat packet parameters are expected to be one of the following: (a) an integer, (b) a floating point, (c) a boolean, or (d) a character string.

A valid integer consists of an optional sign followed by a string of digits. A valid floating point number consists of an optional sign, a string of digits which may contain a decimal point, an optional **e** or **E**, an optional sign, and an optional integer. For example the following qualify as valid floating point numbers: `1200`, `-45.3E-6` and `.01234e10`. Acceptable boolean entries for true are limited to: **true**, **1**, **t**, **y**, and **on**. Similarly, acceptable boolean entries for false include: **false**, **0**, **f**, **n**, and **off**. All leading and trailing white space is removed from any character strings. Comments on character string lines are also removed.

If a numerical value describes a particular parameter, the specific units of measure are explicitly assumed by MUVES. For specific information on the threat types and corresponding threat packets refer to the documentation for each approximation method.

3.2.2 Supplemental Threat Files

For the several classes of threats already incorporated into MUVES, supplemental interpolation tables relating one or more parameters are also required to describe the threat. Any supplemental files containing interpolation tables must be formatted according to the rules discussed in Section 3.1, **Input File Syntax**. Examples of supplemental files for KE threats will be provided in this section to show the type of information which may be considered part of a complete threat specification.

```
#       Interpolation table of striking velocity data
#       for sample_ke threat

#       Striking velocity as a function of range

!       6

#       range   striking velocity
        3000          1708.0
        2000          1800.0
        1500          1900.0
        1000          1905.0
        500           1910.0
        0             2000.0
```

Figure 8. A sample kinetic-energy "range" file

For every KE projectile, the distance between the weapon and the target has been determined to affect the performance of the projectile. Thus, to determine the striking velocity of a KE threat, MUVES uses a "range" file. Figure 8 provides a sample "range" file for the sample_ke threat.

A "range" file is formatted as a one-dimensional table. The sample file in this figure contains six range/striking velocity pairs. All ranges and striking velocities in this file must always be in meters and meters per second, respectively. When making a threat selection through the user interface, the user

interface will prompt for the range from the threat to the target (*i.e.*, the target-threat range). The striking velocity for the specified range will be determined via a direct table look-up or interpolation from values in this table. Thus, the analyst must specify the target-threat range in meters.

Another file that may be required to describe the capabilities of a KE projectile is a "perf" file. This file describes the single-plate Rolled Homogeneous Armor (RHA) normal thickness perforation capability of the projectile (in mm) as a function of the projectile's obliquity with the target (in degrees) and striking velocity (in meters per second).

```
#       Interpolation table of perforation data for
#       sample_ke threat

#       Single plate RHA normal perforation in mm as a function
#       of obliquity in degrees (X margin) and
#       striking velocity in m/sec (Y margin)

!       24      6

        1708.   1800.0  1900.0  1905.0  1910.0  2000.0
5.0     214.0   229.0   244.2   259.6   275.3   291.0
10.0    211.5   226.4   241.4   256.7   272.1   287.7
15.0    207.5   222.1   236.8   251.8   266.9   282.2
20.0    201.8   216.0   230.4   244.9   259.6   274.5
25.0    200.3   221.6   222.2   236.2   250.4   264.8
30.0    186.0   199.1   212.3   225.7   239.3   253.0
35.0    176.0   188.3   200.8   213.5   226.3   239.3
40.0    164.5   176.1   187.8   199.7   211.7   223.8
45.0    151.9   162.6   173.4   184.3   195.4   206.6
50.0    138.1   147.8   157.6   167.5   177.6   187.8
55.0    123.2   131.9   140.6   149.5   158.5   167.6
60.0    107.4   114.9   122.6   130.3   138.2   146.1
62.0    100.8   107.9   115.1   122.4   129.7   137.2
64.0    94.2    100.8   107.5   114.3   121.1   128.1
66.0    87.4    93.5    99.7    106.0   112.4   118.8
68.0    80.5    86.1    91.8    97.6    103.5   109.4
70.0    73.5    78.6    83.9    89.1    94.5    99.9
72.0    66.4    71.0    75.8    80.5    85.4    90.3
74.0    59.2    63.4    67.6    71.8    76.2    80.5
76.0    52.0    55.6    59.3    63.1    66.8    70.7
78.0    46.7    47.8    51.0    54.2    57.4    60.7
80.0    37.3    39.9    42.6    45.3    48.0    50.7
81.0    0.0     0.0     0.0     0.0     0.0     0.0
90.0    0.0     0.0     0.0     0.0     0.0     0.0
```

Figure 9. A sample kinetic-energy "perf" file

Figure 9 contains a "perf" file for our sample_ke threat. In our example, this two-dimensional table is defined by twenty-four obliquities and six striking velocities. The syntax for this file conforms to the rules outlined in Section 3.1, **Input File Syntax**.

3.3 Targets

Analysts must specify several target-related files. These files can be organized into four categories. *First*, there are four files necessary for ray-tracing-related functions. These files describe (a) the host to be used for ray-tracing, (b) the target description, (c) the mapping between the target description identifiers and MUVES component names, and (d) target geometry information. These four files are stored together in a directory which an analyst names as his target. Specifying a target for an analysis equates to selecting this directory and its files.

Second, two optional ray-tracing-related files may be specified. A *view file* may be used to specify the shot pattern for ray-tracing the target description. During an analysis, a *reusable ray file* may be created to store information on each ray-trace to be used as ray-tracing inputs for a subsequent analysis.

Third, there are two files which should be specified for target components. The *component properties file* supplies any component property values relevant for the analysis being performed. A *component category map file* maps the components in the target to the categories recognized by the approximation method being used. This file is required for all analyses. The physical interactions for all components in a particular component category, including damage, are evaluated in the same fashion.

Fourth, analysts must specify two files (and sometimes a third file) for evaluating and assessing the damage resulting from the target-threat interactions. A *damage assessment expression file* lists the contributions of components or target subsystems to the overall functionality of the target in a given context. A *system definition file* describes the target subsystems in terms of the critical components required for continued performance of those subsystems and is not always required. Finally, a *damage evaluation selection file* must be used to select the name of the Evaluation Module desired for assessing remaining component utility for every critical component category. Evaluation Modules are usually specific to an approximation method; however, some modules may be used by more than one methodology.

For each target-related file that must be provided, a portion of a sample file for a simple fictitious tank is supplied as an illustration. A complete set of sample files is distributed with MUVES so that analysts can manipulate copies of these files and make analysis runs with them. These files are stored in the "samples" project directory.

3.3.1 Ray-tracing host file

The ray-tracing host selection file, "host" file, lists the host that can be used for target geometry ray-tracing. Ray-tracing on a remote machine can be done only if *all* target geometry and related files are copied to the remote machine. This insures the same target geometry is used by all remote machines.**

** Although more than one host may be listed in the "host" file, the underlying MUVES software does not support ray-tracing on multiple machines at the time of writing.

```
#       Ray-tracing host file
local
```

Figure 10. A sample "host" file using the local host

For efficiency purposes, if the target geometry ray-tracing is performed on the same machine on which the analyst is running MUVES, then **local** should be specified in the "host" file. Figure 10 provides a sample "host" file specifying **local** ray-tracing.

```
#       Ray-tracing host file using a remote slave
worm
```

Figure 11. A sample "host" file using a remote host

Figure 11 provides a sample "host" file naming a computer system, *i.e.*, worm, for remote slave ray-tracing. The MUVES system administrator must configure a corresponding "hosts" file for each non-local host which may be specified in this file. Analysts should check with their local MUVES system administrator for a list of non-local hosts or for adding non-local hosts to the "hosts" file.

3.3.2 Target Description

Computer models of targets are used to simulate target-threat interactions in MUVES. At this time, the only geometric modeling method supported by MUVES is the BRL-developed Multi-device Graphics EDitor (MGED).‡ MGED is an interactive editor for solid models that is based on Combinatorial Solid Geometry techniques [9,10]. With MGED, designers have the capability to build, view, and modify a target model by interactively manipulating a graphical representation of an image on a graphical display. The file containing a geometric description in MGED typically has a name with a ".g" suffix.

3.3.3 Region Map File

For targets developed with MGED, a region map file associates the component identifiers in the target description with MUVES component names. A component identifier is either the MGED region identifier; or in the cases of air regions (where the MGED region identifier is 0), it is the negative of the air space code [9]. Component names in the region map file are used in other MUVES files, including the component properties file, the component category map file, the damage assessment expression file, the system definition file, and the damage evaluation selection file. Analysts may create any name for a component provided: (1) it does not contain embedded tabs; (2) it is not quoted; (3) it does not contain just digits (*e.g.*, 742); and (4) it does not begin or end with spaces. In the case of a component name containing all

‡ MUVES is designed so that different modeling methods could be interfaced.

digits, it would be impossible to create valid damage assessment expressions. MUVES also will automatically strip off white spaces which begin or end a component name. It is also important to note that these names are (upper and lower) case sensitive.

```
#       Region Map File

#       Air
crew air                -2
ammo air                -3
engine air              -5

#       Armors
upper glacis            104
turret front            205
armor                   100:103 110 115 160 165 206:211

#       Suspension
track left              500
track edge lt           501
track right             502
track edge rt           503
idler left              510
idler right             517
sprocket left           511
sprocket right          518
first wheel lt          505
first wheel rt          512
other wheels            506:509 513:516
```

Figure 12. A sample MGED region map file

Each region map file has a format that is dependent on the geometric method used to develop the description. A sample file for a target developed with MGED is provided in Figure 12. Each component name in a region map file for a MGED description must be followed by a tab and a list of component identifiers. Identifiers corresponding to a component may be given individually (*e.g.*, `-2`, `104`, `205`, *etc.*), specified as a range (*e.g.*, `100:103`, `206:211`, *etc.*), or a combination of individual and range components. If identifiers are specified as a range, the range includes both the first and last identifiers as well as all identifiers numerically in between. The first identifier supplied for a range, *i.e.*, the lower bound, must not exceed the second, *i.e.*, the upper bound. If necessary, the same component name can occur on multiple lines; however, no identifier can belong to more than one component. In this example, all MGED regions with air space code 2 (*i.e.*, `-2`) would be mapped to the MUVES component name `crew air`. Similarly, all MGED regions having identifiers 506, 507, 508, 509, 513, 514, 515, or 516 would be mapped to the MUVES component name `other wheels`.

3.3.4 Geometry File

Ray-tracing on a remote machine is accomplished by copying *all* target geometry and component information to the remote machine. The "geometry" file contains the geometric method used to develop the target description, the names of the files that need to be copied, and method-specific information. For example, Figure 13 is a "geometry" file for a tank description (*i.e.*, `tank.g`) developed using the MGED method. In this file, the method used to develop the target description must be specified first. For targets

```
#       Target database description file

mged                    # Slave geometry method

# The following pairs of keywords and values will satisfy
# requests from the mged-slave:

region_map      comp    # local pathname of region->component
                        # map file
mged_database   tank.g  # local pathname of solid-model
                        # geometry file
root_object     compartment     # name of top-level object in
                                # solid-model geometry file
thin_tolerance  1.0e-5  # eliminate target gaps and traces
                        # through modeled target components
                        # thinner than the specified threshold
```

Figure 13. A sample "geometry" file

built with MGED, **mged** (in lower-case) should be used. Following the identification of the geometric method, MUVES expects a series of keywords and values on subsequent lines in the file. Each keyword and its value must be on a line by itself.

For the **mged** method, the following four keywords and their corresponding values are supported: (a) **region_map** *filename*, (b) **mged_database** *filename*, (c) **root_object** *string*, and (d) **thin_tolerance** *float*. The **region_map** keyword specifies the name of the file that contains a mapping of the identifiers in the target description to MUVES component names. The **mged_database** keyword specifies the filename of the MGED target description. This filename will typically have a ".g" suffix. The **root_object** is the object name in the MGED target description that contains the relevant components; it is normally a so-called group node [9]. The **thin_tolerance** is the critical ray tracing threshold to eliminate geometric modeled gaps and traces through target components which are thinner than the specified threshold. If **thin_tolerance** is not specified, the default tolerance is 1.0e-5. Setting the tolerance to 0 will disable the elimination of thin gaps and component traces.

All filenames specified in the "geometry" file must be in the same directory as the "geometry" file. To insure these files are kept in the same directory (UNIX), filenames specified in the "geometry" file may not start with '/' or '../'.

3.3.5 View File

An analyst may specify all ray-tracing operations through the user interface. However, if a particular specification is to be used more than once, it may be convenient to put this information into a file and reference that file when setting up an analysis. Such files are called *view files* and produce the same results as specifying the ray-tracing operations through the user interface. This is a convenient way to avoid the work of continually specifying the same set of parameters for various analyses.

For convenience, the term shot will be used to refer to the combination of location and direction which defines a trace through the target. The MUVES shot specification language provides a flexible means for specifying desired shots, while making it convenient to specify the types of viewing operations analysts use

most frequently. The shots are defined using a set of keywords and associated arguments. Each keyword must appear on a separate line, and all arguments to that keyword must follow on the same line. Comments are permitted in the view file and follow the standard MUVES input file convention that lines beginning with a '#' character and the remainder of lines containing a tab followed by a '#' character are considered comments.

The following table shows the keywords available for defining the viewing parameters in an analysis. The keyword indicates what is to be defined on the line, and further arguments give the definition for that item. As can be seen in the table, some of these arguments are optional and some are not. In these definitions, the letter *r* in an argument list represents a real number, while the letter *n* represents an integer; if *string* follows a keyword, then it indicates a character string is expected.

TABLE 2. Shot Definition Keywords

Keyword	Required Arguments	Optional Arguments
units	*string*	
shot	**direction** *r r r* *or* **asel** *r r*	**xys** *r r r* *or* **origin** *r r r* *or* **hv** *r r*
group		*string*
endgroup		
grid	**csize** *r r*	**extent** *n n* **random** **seed** *n*
gauss	*n* **sigma** *r r*	**seed** *n*

TABLE 3. Valid Unit Names

Name	Symbol
feet	**ft**
inches	**in**
meters	**m**
centimeters	**cm**
millimeters	**mm**
microns	

In MUVES, all shot specifications are considered to be in millimeters unless stated otherwise. An analyst may change the units within a view file using the **units** keyword followed by a recognized unit name or symbol (Table 3). All values specified in the view file will be treated in the appropriate units. However, the **units** keyword only affects shot specifications which follow this keyword. A good practice is to place the **units** keyword and its value as the first non-comment line in a view file.

To specify a single shot, one uses the keyword **shot** followed by the parameters for that shot. The direction of the shot may be specified using either a direction vector, or an azimuth and elevation. One specifies a direction vector using the **direction** argument followed by the three components of the vector. One specifies an azimuth and elevation using the **azel** argument followed by the azimuth and elevation. One of these direction specifiers *must* appear in the shot specification. Note that **azel** and **direction** are mutually exclusive; only one of these may appear in a shot specification. The position of the shot may also be specified in one of three ways. An absolute point may be specified using **xyz** followed by the coordinates of the point. A location in the plane perpendicular to the viewing direction may be specified using **hv** followed by the desired horizontal and vertical offsets from the target origin. Both of these will produce a shot from outside the target space going through the desired location. The keyword **origin** may be used to specify the absolute origin of the shot in the target space. The difference between this and **xyz** is that **xyz** produced a shot guaranteed to start outside the target space and go *through* the specified point, thus hitting any portions of the target which may mask that point; while **origin** causes the shot to ignore anything behind the point. This distinction is significant for some applications. One may omit all position specifiers; the result will be a shot going through the target origin (0,0,0). The position keywords are also mutually exclusive, *i.e.*, **xyz** and **hv** may not appear in the same shot definition.

Therefore, if one wished to specify a shot from 30 degrees azimuth and 0 degrees elevation, and wanted the shot to go through a point corresponding to some item on the target, measured at (312.4, 102.7, -42.5), one would type the following line:

```
shot azel 30 0 xyz 312.4 102.7 -42.5
```

In MUVES, all shots are part of something called a *shot group*, which may contain a single shot or an arbitrarily large collection of shots. This concept is synonymous with the term *view*, since it defines a desired collection of shots for investigating a target. Both terms are used to refer to any collection of shots which the analyst wishes to manipulate *as a unit*, whether for visual display, statistical analysis, or any other desirable operation. The term *view* is used frequently when discussing postprocessors.

In order to combine more than one shot into a shot group, MUVES provides the **group** keyword. This keyword initiates a group of shots, which is terminated using the **endgroup** keyword. The **group** keyword may have a label following it to make referencing more convenient, though this is not required. Any shot specifications which appear between the **group** and **endgroup** keywords are considered to be part of a single shot group and will be listed together in the final results file.

For purposes of further discussion, a shot which does not appear within a **group**/**endgroup** pair is considered to be a shot group containing one member. Thus, every shot in a view file is part of some group.

In the example below, the group labeled **spray** contains five shots going through the same point from different directions. These shots will be treated as a single view by the user interface and all postprocessors. The lone shot following this group will be treated as a separate view; it is considered a group even though it is not bracketed by the **group** and **endgroup** keywords.

```
# Several shots through the same point
group spray
shot  azel 0 0  xyz 581 57 106
shot  azel -10 0  xyz 581 57 106
shot  azel 10 0  xyz 581 57 106
shot  azel 0 -10  xyz 581 57 106
shot  azel 0 10  xyz 581 57 106
endgroup

shot azel 30 10  hv 300 55
```

Figure 14. Example of Shot Grouping

Although the grouping facility allows the analyst to specify large arbitrary collections of shots, it would be rather tedious to create regular patterns of shots in this manner. Recognizing that analysts would rapidly get tired of specifying each desired shot separately, MUVES provides the ability to overlay patterns on a shot. A pattern is a collection of shots offset from a defined location and direction in some predictable fashion. The **shot** keyword is used to define the location and direction, while a pattern definition provides the specific method of offsetting the patterned shots. This offset may affect the location, the direction, or both. After a pattern is overlaid on a shot, the shot is not individually ray-traced by MUVES; only the shots produced by the pattern are ray-traced.

Using patterns does not prohibit inclusion in a shot group; any or all shots in a shot group may have patterns overlaid on them.

A grid is the most common pattern used by analysts for examining a target from a particular viewing direction. This pattern divides the viewing plane into rectangular sections called "cells", each of which

has a single shot traveling through it; all of the shots in the grid are parallel.

To specify a grid pattern, one uses the keyword **grid** followed by **csize** and a horizontal and vertical dimension for the cells in the grid. There are two optional arguments as well. Adding the word **random** causes the firing point to be randomly located within each cell; if this is not specified, the shot is fired through the center of the cell. In order to initiate reproducible random behavior, the word **seed** followed by an integer will cause the random number generator to begin with that seed. The default seed behavior is to use 1 as the seed for the first view in a session and use an inherited seed for all succeeding views; the seed used for each view will be recorded in the final results file. By default, the grid extends over the entire target. If one wishes to limit the size of the grid, one should use the argument **extent** followed by the desired number of cells in the horizontal and vertical directions.

One peculiarity of the gridding parameters must be mentioned and illustrated. Normally, the analyst desires the grid to be aligned such that the center of the grid lies in the center of a cell. The default gridding condition (covering the entire target) will produce this result. However if the keyword **extent** is used to specify the number of cells to be gridded the behavior may be different. If an odd-by-odd number of cells is specified, the center of the grid will still lie in the center of a cell. If an even number of cells is chosen (for either direction), the center of the grid will lie along the cell boundaries; that is, it will be offset one half-cell from the center of a cell.

An illustration may help clarify this point. In the drawings below, the "X" represents the location of the defining shot for a view. The first drawing shows a 2-by-2 grid pattern centered on a defining shot. The shot lies at the junction of the four cells. However, in the second case, we have a 3-by-3 grid, and the shot lies in the center of the middle cell.

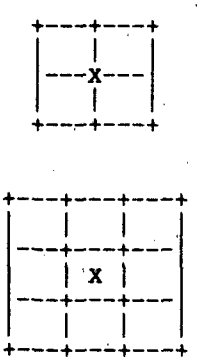


Figure 15. Center of Grid

Finally, MUVES provides a pattern which is not commonly used in vulnerability analyses today but may prove useful nonetheless: the Gaussian pattern. This allows the analyst to specify a number of shots to be randomly located in a bivariate Gaussian distribution (*i.e.*, a bivariate normal distribution) around the defining shot. This is specified using the **gauss** keyword followed by the number of shots desired for the pattern. The horizontal and vertical standard deviations for the distribution are given after the argument **sigma**.

Given the wide variety of selections available, most analysts will wish to create a standard view file and use that file when setting up their analyses. To illustrate the language above, we will develop a view file containing the standard VLD pattern of shots, 7 views in 30 degree increments around the left side of the target with 4-inch grid cells. The first shot would look like this:

```
shot azel 0 0
```

This is a single shot going through the target origin (the default shot location) from 0 azimuth and 0 elevation. Now we want to overlay a 4-inch grid pattern on that shot, so we add the following on a separate line:

```
grid csize 4 4
```

This defines a grid with 4 by 4 for the cell size. To ensure that these cells will be in inches, the unit specification should be in the resulting view file. The grid will automatically be sized to cover the entire target with cells of that size. Note that all shots will be in the center of each cell unless we add **random** to the **grid** specification, as in the following combined example:

```
shot azel 0 0
grid csize 4 4 random
```

Adding this keyword will cause MUVES to alter each shot location by random horizontal and vertical offsets. The **random** keyword may be followed by an optional keyword **seed** with an integer parameter specifying a seed for the random number generator. If no such parameter is given a default seed will be used; each view will inherit its seed from the previous view. In addition, the seeds will be recorded with the final results for later reference. In general, it is better to use the default seed, since that will help avoid correlation problems with the random numbers produced using the random number generator. The ability to specify a seed is included so that analysts can exactly reproduce a particular view when there is some question about the results for that view. Note that, if more than one threat is combined with a given set of views *in a single run*, the views will exhibit different random behavior for each threat. To avoid this problem, the analyst may wish to specify a seed for the first view in the set, allowing the succeeding views to inherit their seeds from the first view.

Returning to our example, we now wish to add six more views, each with the same grid pattern, so our view file would look like this:

```
# Standard 7 views (4-inch grid cells)
units inches

shot azel 0 0
grid csize 4 4 random seed 13579

shot azel 30 0
grid csize 4 4 random

shot azel 60 0
grid csize 4 4 random

shot azel 90 0
grid csize 4 4 random

shot azel 120 0
grid csize 4 4 random

shot azel 150 0
grid csize 4 4 random

shot azel 180 0
grid csize 4 4 random
```

Figure 16. Standard View file for VLD Analyses

Note that we do not need the **group** and **endgroup** keywords, since shots and their associated overlays are separated by default, and we want each of these grids to produce a separate view in the final results file. Note also that specifying a seed for the first view ensures that every time this view file is used for a given target, the same set of shots will be produced. Once created, this view file could be moved into the MUVES input file hierarchy via the user interface (see Section 4.2.1.2, **User Interface, Analysis Configuration - Input Selection**). It could then be referenced during any analysis session.

A sample view file is provided with the MUVES distribution in the "samples" project directory (see the **User Interface** section, **Analysis**).

3.3.6 Reusable Ray File

A reusable ray file is both a MUVES input and output. When making a MUVES run, analysts have the ability to capture all ray traces in binary format to a file which may be used for subsequent analyses. The purpose of reusable ray files is to speedup target geometry interrogation when the same samples of the target are being used repeatedly in the course of a study.* These files can be especially useful when ray tracing is performed by a remote machine or when it is desired to avoid lengthy target geometry preparation delays. Since the files are stored in binary format, they are not guaranteed to work across different machine architectures.

* Preliminary use of reusable ray files has decreased subsequent analysis run times by a factor of five. Analysts should expect this factor to vary based on the target description, view information, and machine architecture.

Factors which produce unique rays include the target and view information. Due to the dynamic nature of the threat-component interaction simulation, ray traces obtained from one analysis, however, might not be the traces needed for a subsequent analysis, even if the target and view are identical. For instance, if shot patterns with randomly fired rays are specified, ray-tracing inputs for different locations may be requested.

The MUVES reusable ray option is designed for efficiency purposes to reuse traces from the file in the order that the traces were recorded. Analysts should set-up analysis runs so that rays are being used from the file in the order that they were created. This can be easily accomplished by using view files or specifying shots in the same order for each analysis. If rays are not being used from the reusable ray file in the order that they were created, matching rays will be found; however, they will be located in a very inefficient manner.

If a requested ray cannot be found in the reusable ray file then MUVES will revert to actual ray-tracing for *all* subsequent ray requests. Thus, the reusable ray file should be created during an analysis run that is expected to produce the most complete set of ray traces. When creating a reusable ray file, the same set of ray traces should not be created multiple times since these files can be quite large. For instance, creating a reusable ray file when running more than one threat can waste a tremendous amount of disk space.

3.3.7 Component Properties File

A component properties file supplies information about the physical properties of components listed in the region map file.† Properties of a component influence the outcome of the target-threat physical interaction and evaluation of the damage. Valid properties are defined with each approximation method. For a particular approximation method, default property values may be defined for components in a certain component categories.

For certain approximation methods, a component properties file may not be required. For example, a component properties file is not required if the default component properties for the approximation method are sufficient for the target being analyzed.** Typically, it is also unnecessary to specify component properties for every component listed in the region map file. Analysts should refer to the documentation on the approximation method for specific information on default component properties and specifics on the method's Interaction Modules.

Figure 17 contains a component properties file for a target being analyzed with the "compart" approximation method, *i.e.*, MUVES implementation of the Compartment model. According to this

† The component properties file may also contain component property values for components which are not in the target being analyzed. This feature may be useful for studies where one component properties file could be shared for several targets.

** Although it would be acceptable not to specify a component properties file, analysts may find it desirable to provide a zero-length file or a file containing a comment that the default properties are being used.

```
#       Component properties file

#       Armor
upper glacis    DENSITY                 7.7641
                THICKNESS_FACTOR        1.0

armor           DENSITY                 7.7641
                THICKNESS_FACTOR        1.0

#       Armament
guntube         DENSITY                 7.7641
                THICKNESS_FACTOR        1.0
                CALIBER                 105.0

#       Other
driver          DENSITY                 1.099
                THICKNESS_FACTOR        1.0
fuel            DENSITY                 0.7972
                THICKNESS_FACTOR        1.0
fuel line       DENSITY                 7.7641
                THICKNESS_FACTOR        0.1
bulkhead        DENSITY                 7.7641
                THICKNESS_FACTOR        1.0
computer        DENSITY                 2.7695
                THICKNESS_FACTOR        0.5
propellant      DENSITY                 1.6900
                THICKNESS_FACTOR        0.95
warhead         DENSITY                 1.65
                THICKNESS_FACTOR        0.80
```

Figure 17. A sample component properties file

component properties file, the target has numerous components, such as, an `upper glacis`, `armor`, `fuel line`, *etc.*. Not all components listed in the component properties file need to exist in the target; however, all property names (*e.g.*, `DENSITY` and `THICKNESS_FACTOR`) must exist in the list of component properties defined for the approximation method. The list of component properties for a particular vulnerability assessment methodology will be provided with the description of each approximation method.

All component names and property names must be separated from property names and property values, respectively, by one or more tabs. If a component is described by more than one property, only one property and its value may be given on a line. *All* property names must be preceded by one or more tabs.

3.3.8 Component Category Map File

A component category map file lists the component categories that an analyst has assigned to component names. The component category map file must be specified for every combination of target and approximation method being analyzed under MUVES.

Figure 18 is an example of a component category map file for a target for analysis with the "compart" approximation method. Notice that this sample file has several component categories, *e.g.*, `crew compartment`, `armor`, `fuel`, *etc.* For each category, zero, one or several components may be specified.

```
#       Component category map file for use
#       with the "compart" approximation method

#       Every component in the component category
#       map file must be mapped to one of the available
#       component categories in the Interaction Module Table
#       for the approximation method. The category
#       determines how the threat-component interaction
#       is modeled.


#       Category                Component

crew compartment                crew air

armor                           armor
                                turret front

fuel                            fuel

other                           driver
                                gunner
                                loader
                                commander
                                propellant
                                fuel line
                                fuel filter
                                bulkhead
                                engine
                                transmission
                                computer
                                other wheels

target gap                      MUVES target gap

exit paint                      MUVES exit paint
```

Figure 18. A sample component category map file

Thus, a category may be listed with no matching components. Analogous to the component properties file format, component categories must be separated from component names by one or more tabs. Only one component name, preceded by one or more tabs, may be supplied per line. For instance, the **armor** category contains `armor` and `turret front` components.

Every component in the region map file *must* be assigned to one category in the component category map file. To facilitate analyses which could share a common component category map file, the component category file may contain components which are not listed in the region map file.‡ To simulate the physical interaction, all categories in the component category map file must exist in the Interaction Module table supplied by the approximation method designer.

‡ For example, in a concept vehicle study, it might be advantageous for several target variations to share a common component category map file.

In addition to the explicitly modelled components in the target, MUVES also defines two special components — **MUVES exit paint** and **MUVES target gap**. **MUVES exit paint** is automatically supplied as the last zero-thickness component on a threat path. **MUVES target gap** is automatically supplied to fill in voids in the target description. These two components *must be* included in the component category map file.

The consequences of the physical interaction for components in the same component category are produced and evaluated in the same manner. The list of allowable categories will be discussed with the approximation method.

3.3.9 Damage Assessment Expression File

A damage assessment expression file lists the contributions of components or target subsystems to the overall functionality of the target in a given context. Functionalities are expressed in terms of the target's ability to perform its desired mission function(s) in the given environment. Thus, these expressions are in terms of fractional remaining functionalities (FRFs). Functionality expressions may be created by combining any of the following:

- a constant (*e.g.*, 0.5)
- a component name (*e.g.*, "turret front")
- a system name (*e.g.*, "engine power")
- a unary-operator expression using a not, an absolute value, or a boolean operator followed by an expression (*e.g.*, ~"track left" implying not "track left", @"guntube" implying the absolute value of the "guntube", ?"track" implying a boolean evaluation for the "track")
- expressions separated by the binary operators: and, exclusive-or, or, maximum, minimum, product, sum, difference (*e.g.*, "master power" & "engine power" is an AND expression of two system components)
- an expression enclosed in parentheses

Any component and system names containing tabs and spaces in these expressions must be enclosed in double quotes. Component and system names containing any unary operators, binary operators, or parentheses must also be enclosed in double quotes.

The logical and mathematical operators that may be used to build expressions are provided in Figure 19. The precedence of these operators is given in this figure. Parentheses may be used to clarify or modify the built-in precedence rules. To aid in understanding each of these operators, sample expressions are provided.

Unary Operators

absolute-value	@
boolean-value	?
not	!, ~

Note: If the boolean operand is greater than 0, then the boolean value is 1; otherwise, the boolean value is 0.

Binary Operators

and	&
difference	-
exclusive-or	^
maximum	>>
minimum	<<
or	|
product	*
sum	+

Operator Precedence (decreasing order)

Unary operators (evaluated right to left)

! ~

@

?

Binary operators (evaluated left to right)

&

^

|

<< >>

*

+ -

Figure 19. Unary and binary operators

Figure 20 contains a damage assessment expression file that contains three damage assessment expressions. As stated previously, each expression evaluates the functionality based on the damage states specific to the environment and mission. The first damage assessment expression is for a target in a `typical` environment with a `firepower` mission. The second is for the same target in a "`european offensive`" environment with a `mobility` mission. The third expression is for the target in a "`european defensive`" environment with a `mobility` mission.

```
#       Damage assessment expression file

typical
firepower
"crew air"&
"ammo air"&
"fuel tank"&
guntube&
warhead

"europe offensive"
mobility
(0.4*(! "master power" & "engine power")) +
(0.4*("master power" & ~ "engine power"))

"europe defensive"
mobility
 ((0.3*("master power" >> "engine power")) -
  (0.3*("ammo air" << "crew air")))
| (0.2*"fuel supply system")
```

Figure 20. A sample damage assessment expression file

Newlines are required after both the environment and mission identifiers. As with other files, newlines used elsewhere in the file are ignored. White space is ignored except if the white space is escaped inside the name of an identifier (*e.g.*, `track\ left`) or if the identifier containing white space is enclosed in double quotes (*e.g.*, `"track left"`). Identifiers may contain digits but must not qualify as constants, *i.e.*, they may not consist only of digits and an optional decimal point.

To understand the interpretation of these expressions, let us consider several theoretical possible values for the components in the first expression. Suppose that all components are undamaged, *i.e.*, each has a FRF of 1. Combining values of 1 with the AND operator is equivalent to multiplying these values together. In this example, the overall expression value would be 1. Thus, as one might expect, the target is fully functionally for its mission. Suppose, however, that the guntube was damaged and is only 50% functional. Multiplying the four 1 values and one 0.5 value results in a remaining functionality of 0.5. It is important to note that multiplying together functionality values is equivalent to the *survivor rule* on loss-of-function values.*

In the second expression for the `"european offensive"` environment, suppose the `"master power"` is only 75% functional and the `"engine power"` is 25% functional. The not of `"master power"` evaluates to 0.25. Consequently, `(! "master power" & "engine power")` evaluates to 0.0625. Similarly, `("master power" & ~ "engine power")` evaluates to 0.5625. Multiplying each of these

---

* The *survivor rule* used in VAMP states that the combined loss of function for n critical components (LOF_n) is given by:

$$LOF_1 = LOF_1$$
$$LOF_2 = [\, 1 - LOF_{(1)} \,]\, LOF_2 + LOF_1$$
$$LOF_n = [\, 1 - LOF_{(n-1)} \,]\, LOF_n + LOF_{n-1}$$

expressions by 0.4 and adding the results gives a FRF of 0.25 (from evaluating 0.025 + 0.225).

The third expression in the damage assessment expression file is for a target in a "european defensive" environment with a mobility mission. To demonstrate the evaluation of this expression, the following FRFs will be assumed: (a) "master power" equals 0.75, (b) "engine power" equals 0.25, (c) "ammo air" equals 0.80, (d) "crew air" equals 0.50, and (e) "fuel supply system" equals 0.60. To evaluate (0.3*("master power" >> "engine power")), one must first take the maximum of "master power" and "engine power". In this example, the maximum value is 0.75, and the expression evaluates to 0.225. To evaluate (0.3*("ammo air" << "crew air")), one must take the minimum of "ammo air" and "crew air". In this example, the minimum value is 0.5. Thus, the expression evaluates to 0.15. Taking the difference of these two expressions results in a value of 0.075 (from evaluating 0.225 − 0.15). Similar to earlier examples, the expression (0.2*"fuel supply system") evaluates to 0.12. Overall probability of the "or" of these terms is readily evaluated by DeMorgan's (survivor) rule: 1.0 − (1.0 − 0.075) * (1.0 − 0.12.) which equals 0.186.†

3.3.10 System Definition File

A system definition file describes the target subsystem operation in terms of critical components. Critical components are components required for the continued performance of the selected mission functions (*e.g.*, mobility, speed, firepower, personnel loss, *etc.*) in a particular environment (*e.g.*, nighttime in a forest, rough roads, typical, *etc.*). In MUVES terminology, target subsystems are defined by relationships between groups of target components. If the damage assessment expression file refers only to component names and does not refer to any target subsystems, a system definition file is not required. If there are no target subsystems then a zero-length file or a file containing just comments and/or white-space may be used if the analyst prefers to use a system definition file. If the damage assessment expressions had contained one or more target subsystems, then the target subsystem(s) would have to be defined in the system definition file.

To show the format of a system definition file containing target subsystems, Figure 21 contains a sample file with three target subsystems: the "master power", the "engine power", and the fuel\ supply\ system. Of these, "engine power" is perhaps the easiest system to understand; however, it only uses a small subset of the available operators. "Master power", on the other hand, uses most of the available operators, and the fuel\ supply\ system just appears complicated since escaped white space is used instead of enclosing identifiers containing white spaces in double quotes. Notice each system is followed by an equals-sign and an expression.

System definition expressions use the same syntax described for damage assessment expression files (refer to Section 3.3.9, **Damage Assessment Expression File**). All component and system names

† If an exclusive-or had been used instead of an or, the expression would reduce to (0.075 + 0.12 − (0.075 * 0.12)) * (1.0 − (0.075 * 0.12)) which equals 0.184.

```
# (Fictitious) system definition file

# Definition of master power:

"master power" = 0.5 >>  # Minimum value of 0.5 for this system
        !(  ( "engine power"
            & "cable 2w108"
            & "voltage regulator"
            & ~ "hull networks box"
            & "engine disconnect panel"
            & ("cable 3w102-1" ^ "cable 3w102-2")
            * "cable 2w157"
            + "cable 2w158"
            ) | ? batteries
        )
        & "terminal boards"
        & @"cable 2w154-2w155"

# Definition of engine power:

"engine power" =  "fuel supply system"
        & "driver's master panel"
        & "cable \"2w104\""
        & "hull networks box"
        & "electronic control unit"
        & compressor


# Definition of fuel supply system (Note escaped symbols are used
# rather than quoting the entire identifier.):

fuel\ supply\ system =
          left\ rear\ fuel\ tank
        & right\ rear\ fuel\ tank
        & rear\ fuel\ tank\ interconnect
        & (f.line\ \-\ l.r.\ tank\ to\ tee |
           f.line\ \-\ r.r.\ tank\ to\ tee)
        & f.line\ tee
```

Figure 21. A sample system definition file

containing tabs and spaces may be enclosed in double quotes, or use escaped tabs and spaces (*e.g.*, "`fuel supply system`" or `fuel\ supply\ system`). Identifiers may contain digits, but must not qualify as constants. The unary and binary operators in system definitions are the same as those used in damage assessment expressions. Although newlines are required between system definitions, newlines used elsewhere in the file are ignored.

3.3.11 Damage Evaluation Selection File

A damage evaluation selection file must be used by the analyst to select the name of the Evaluation Module desired for assessing the remaining component utility for *every critical* component category. Non-critical component categories listed in the damage evaluation selection file are ignored. All component categories in this file must be defined in the Interaction Module Table for the particular approximation method. Valid component categories will be discussed with each approximation method. Evaluation of the damage to each component category is performed by the Evaluation Modules. Similar to Interaction

Modules, the Evaluation Modules are unique to the approximation method and threat type. Consequently, valid evaluation modules will be discussed with each approximation method.

```
#       Damage evaluation selection file

# Any component/system in a damage assessment expression file
# causes that component's category to become critical.

# There must be a one-to-one mapping of all critical
# categories to evaluation modules.


# Critical Category             Evaluation Module

ammo compartment respen         ammo compartment respen
crew compartment                crew compartment
engine compartment              engine compartment
fuel                            fuel
gun tube                        gun tube
track                           track
sprocket                        sprocket hub
idler                           idler hub
first roadwheel                 first roadwheel hub
ammo piece                      ammo piece
```

Figure 22. A sample damage evaluation selection file

Figure 22 contains a damage evaluation selection file for the "compart" approximation method. Every non-comment line in this file contains a component category followed by one or more tabs, and an Evaluation Module name. For instance, if the **ammo compartment respen** category is encountered then the corresponding damage is produced by the **ammo compartment respen** evaluation module.

3.4 Underlying Approximation Method Inputs

MUVES relies on Interaction Modules to model the physical interaction and Evaluation Modules to evaluate the damage to components. Certain modules may require inputs which are a function of the approximation method, the target's characteristics (components, component properties and component categories), and/or the threat's characteristics. If required, these inputs are best described as either Interaction or Evaluation Module inputs. Although these inputs are highly dependent on the method, the target, and the threat, this section will outline several common types of files that may be required by the underlying modules. It is hoped that these examples will give the reader a better understanding of the kind of information that may be required. Analysts must refer to the approximation method documentation for specific information.

3.4.1 Interaction Modules Inputs

The Interaction Module for a particular threat type and component category is responsible for invoking the appropriate physical interaction algorithms. Physical interaction algorithms compute parameters required to evaluate damage within critical component categories. Threat-specific interpolation tables, such as "perf" files for KE munitions (see Section 3.2.2, **Threat Information, Supplemental Threat Files**), may be used by the KE Interaction Modules to compute some of these parameters. Parameters from the "initial" file are also used. Provided there are no parameters dependent on both target and threat characteristics, the files discussed up to this point are sufficient for modeling the physical interaction.

For some Interaction Modules other input files that are not specific to a particular threat may be required. As an example, the spall fragments generated when the threat impacts a particular component may be important for an analysis. If so, an Interaction Module may require an input table on the number of lethal spall fragments produced. It is anticipated that the inputs required by the interaction modules can be specified in one-dimensional and two-dimensional tables. If more than one table is required, each table should be labeled with a table tag and included in the same file.

3.4.2 Evaluation Modules Inputs

After the target-threat physical interactions have been computed, damage evaluation is performed on critical components on the threat path. Evaluation Modules perform damage assessments. One way to assess damage is via tables which relate parameter and loss-of-function (LOF) values. In the vulnerability/lethality literature, these tables are typically referred to as damage correlation curves even though the curve may be a step function or a set of connected observations. For instance, a set of damage correlation curves for a particular approximation method may be described according to the threat type, critical component category, mission, and environment. In this case, table tags would be used to separate the curves in this file. However, other approximation methods may not require inputs to assess the damage. In this case, the evaluation modules may not require any inputs. For specific information on the Evaluation Module inputs, analysts should refer to documentation for each approximation method.

3.5 File Formats summarized using Backus-Naur Form (Optional)

The inherent structural properties of threat and target descriptions can be naturally expressed by a context-free grammar such as Backus-Naur Form (BNF). BNF notation is popular for formally specifying syntax and facilitating semantic interpretation. It is particularly useful for describing programming languages and inputs which must be parsed by software programs [12].

For analysts unfamiliar with BNF notation, the salient characteristics of BNF notation will be briefly described, followed by the specific grammar for the input files previously discussed in this section. ***This section is strictly optional and may be skipped without the loss of any information. This information is only provided as a short-hand reference for those readers who feel comfortable with this notation.***

3.5.1 Notation (Optional)

As an example of a BNF grammar, let us consider how simple integer expressions are described using this notation. From elementary math, a positive integer can be described as a sequence of digits preceded by an optional plus sign. Similarly, a negative integer can be described by a negative sign followed by a sequence of digits. Thus, -1, +10, -101, and 0 are clearly valid integers. Using BNF notation, integers may be formally expressed as follows:

integer ::= sign {digit}+

sign ::= '-' | '+' | <empty>

digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

The ::= symbol may be read as "can have the form". Thus, all items on the left-hand side of the ::= symbol are terms which are defined by using one or more symbols from the right-hand side of the ::= symbol, *i.e.*, nonterminals. All symbols on the right-hand side of the ::= symbol may contain any combination of nonterminals and terminal symbols, *i.e.*, symbols which require no further explanation. To facilitate readability, terminal symbols, *a.k.a.*, tokens, are always enclosed by single quotes or angle brackets if the symbol cannot be described by a single entity (*e.g.*, <newline character>). The special empty token, *i.e.*, <empty>, indicates an empty string of symbols is acceptable.

To define a digit, the pipe symbol, | , is used to indicate "or", signifying that only one of the listed symbols must be selected. An "or" sequence where the pipe symbol is used repetitively for consecutive items may be shortened to the delimiting endpoints of the sequence separated by a dash, *e.g.*,

digit ::= '0' - '9' is equivalent to digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'.

BNF notation also uses specialized notation to convey information on the number of optional and required repetitions of symbols. For notational convenience, these structures may be combined to define an arbitrarily complex nonterminal. This notation includes:

{ } indicates that *one* instance of the symbol(s) enclosed in the curly braces must appear

{ }? indicates that *zero or one* instance of the symbol(s) enclosed in the curly braces must appear

{ }* indicates that *zero or more* instances of the symbol(s) enclosed in the curly braces may appear

{ }+ indicates that *at least one* instance of the symbol(s) enclosed in the curly braces must appear

The rules which have been used to define an integer in this section are referred to as production rules. Each production rule is comprised of a nonterminal symbol followed by a ::= symbol and a sequence of nonterminals and/or tokens. In any set of production rules, there is one special nonterminal from which all other nonterminal symbols are derived. This special nonterminal symbol is referred to as the start symbol. In our particular example, integer is the start symbol.

Given a math processing program needs to read in and interpret mathematical expressions involving integers and operators, a complete BNF notation for a simple math processing program might be described as follows:

expression	::=	integer { {WS}+ operation {WS}+ integer }+ {WS}* NL
integer	::=	{sign}? {digit}+
sign	::=	'-' | '+'
digit	::=	'0' - '9'
WS	::=	' ' | <tab>
operation	::=	'+' | '-' | '*'
NL	::=	<newline character>

According to this grammar, 25, -4, -0, 0, and 100000 are valid integers but not valid expressions since there is not at least one operation and another integer. However, note that 1,000 and 5.0 are clearly not valid integers since they contain a comma and decimal point, respectively. Similarly, 1+2–6*4 and 4*1000+2 are valid expressions; however, */1 and 2**5 are not valid expressions due to the ordering of the integers and operations.

As stated earlier in this section, all alphabetic and alpha-numeric strings in the MUVES files are (upper and lower) case sensitive. This must be implicitly assumed in all MUVES BNF grammars since it is nearly impossible to reflect case sensitivity, particularly in cases where a specific string is used across several files. Further, blank lines and comment lines, *i.e.*, lines starting with a '#' character, are acceptable in any MUVES file. The remainder of any line which contains a '#' character immediately preceded with a tab character is also a comment. All comments are automatically skipped over by the MUVES software during processing. However, to reduce complexity, blank lines and comment lines are not explicitly shown in any MUVES file format grammars. The remainder of this section will provide the BNF notation for a subset of the MUVES input files discussed in this section.

3.5.2 Backus-Naur notation for "initial" files (Optional)

```
initial-file      ::=  threat-type NL { threat-packet }*
threat-type       ::=  <name for the threat type>
threat-packet     ::=  packet-name NL { parameter NL }* 'endpacket' NL
packet-name       ::=  <name of threat packet type>
parameter         ::=  integer
                       | floating-point
                       | boolean-value
                       | <character-string ignoring leading and trailing white-space>
integer           ::=  { sign }? { digit }+
sign              ::=  '-' | '+'
digit             ::=  '0' - '9'
floating-point    ::=  { sign }? { digit }+ { '.' }? { digit }* { 'e' | 'E' }?
                           { sign | <space> }? { integer }?
                       | { sign }? { '.' }? { digit }+ { 'e' | 'E' }?
                           { sign | <space> }? { integer }?
boolean-value     ::=  'true' | '1' | 't' | 'y' | 'on'
                       | 'false' | '0' | 'f' | 'n' | 'off'
NL                ::=  <newline character>
```

Figure 23. Backus-Naur notation for "initial" files

3.5.3 Backus-Naur notation for ray-tracing "host" files

```
host-file   ::=  { host-line NL }+
host-line   ::=  <name of the host containing the target geometry data>
NL          ::=  <newline character>
```

Figure 24. Backus-Naur notation for ray-tracing "host" files

3.5.4 Backus-Naur notation for MGED region map files

The following figure contains the BNF notation for a component identifier map file where the target description was built using BRL MGED.

```
mged-map-file   ::=  { comp-line NL }*
comp-line       ::=  comp-name HT ident-list
comp-name       ::=  <name of component (no embedded tabs)>
HT              ::=  <horizontal-tab character>
ident-list      ::=  {WS}* { range | ident } { {WS}+ { range | ident } }*
WS              ::=  <non-newline white-space character>
range           ::=  ident ':' ident
ident           ::=  <MGED ident that maps into this component>
NL              ::=  <newline character>
```

Figure 25. Backus-Naur notation for MGED region map files

3.5.5 Backus-Naur notation for "geometry" files

```
geom-file     ::=  method-line NL {key_line NL}+
method-line   ::=  'mged'
NL            ::=  <newline character>
key_line      ::=  'region_map' filename | 'mged_database' filename
                   | 'root_object' string | 'thin_tolerance' float
```

Figure 26. Backus-Naur notation for "geometry" files

3.5.6 Backus-Naur notation for component properties files

```
comp-file     ::=  { comp-lines }*
comp-lines    ::=  comp {HT}+ { prop-list }+
comp          ::=  <name of the component category (no embedded tabs)>
HT            ::=  <horizontal-tab character>
prop-list     ::=  property {HT}+ prop-value NL
property      ::=  <component property>
prop-value    ::=  <property value for the component>
NL            ::=  <newline character>
```

Figure 27. Backus-Naur notation for component properties files

3.5.7 Backus-Naur notation for damage assessment expressions and system definitions

The underlying structures of damage assessment expressions and system definitions were designed to be similar. The same unary and binary operators in the following figure (as shown in Section 3.3.9, **Damage Assessment Expression File**) are used for these expressions and definitions. These operators must be used to complete the BNF notation. The BNF grammar common to both files is shown followed by the specific BNF notation for each file.

Unary Operators

absolute-value	@
boolean-value	?
not	!, ~

Note: If the boolean operand is greater than 0, then the boolean value is 1; otherwise, the boolean value is 0.

Binary Operators

and	&
difference	-
exclusive-or	^
maximum	>>
minimum	<<
or	|
product	*
sum	+

Operator Precedence (decreasing order)

Unary operators (evaluated right to left)

! ~

@ ?

Binary operators (evaluated left to right)

&

^

|

<< >>

*

+ -

Note: Parentheses may be used to override the built-in precedence rules.

Figure 28. Unary and binary operators

Production rules with non-terminals/terminals on the right-hand side:

```
expression          ::=  constant
                       | component-name
                       | system-name
                       | unary-operator expression
                       | expression binary-operator expression
                       | left-paren expression right-paren
component-name      ::= identifier
binary-operator     ::= and | or | minimum | maximum | exclusive-or | sum |
                        difference | product
unary-operator      ::= not | absolute-value | boolean-value
identifier          ::= quote { symbol }+ quote  | { character }+
character           ::= regular-character | escaped-symbol
regular-character   ::= digit | letter | underscore | period
constant            ::=  digit-sequence { decimal-point }?
                       | digit-sequence decimal-point digit-sequence
                       | decimal-point digit-sequence
digit-sequence      ::= { digit }+
decimal-point       ::= period
escaped-symbol      ::= escape symbol
```

Production rules with only terminals on the right-hand side:

```
NL                  ::= <newline character>
absolute-value      ::= '@'
and                 ::= '&'
apostrophe          ::= '''
boolean-value       ::= '?'
difference          ::= '-'
digit               ::= '0' - '9'
equals-sign         ::= '='
escape              ::= '\'
exclusive-or        ::= '^'
left-paren          ::= '('
letter              ::= 'A' - 'Z' | 'a' - 'z'
maximum             ::= '>>'
minimum             ::= '<<'
not                 ::= '!' | '~'
or                  ::= '|'
period              ::= '.'
product             ::= '*'
quote               ::= '"'
right-paren         ::= ')'
space               ::= ' '
sum                 ::= '+'
symbol              ::= <ASCII text character [octals 040 - 176] >
underscore          ::= '_'
```

Note: If the boolean operand is greater than 0, then the boolean value is 1;
otherwise, the boolean value is 0.

Figure 29. Backus-Naur notation common to damage assessment expressions and system definitions

```
damage-assessment-expression-file   ::=   { damage-assessment-expressions }+
damage-assessment-expressions       ::=   environment mission damage-states
environment                         ::=   identifier NL
mission                             ::=   identifier NL
damage-states                       ::=   expression
```

Figure 30. Backus-Naur notation specific to damage assessment expression files

```
system-definition-file   ::=   { system-definition NL }+
system-definition        ::=   system-name assignment-operator expression
system-name              ::=   identifier
assignment-operator      ::=   equals-sign
```

Figure 31. Backus-Naur notation specific to system definition files

3.5.8 Backus-Naur notation for damage evaluation selection files

```
des-file          ::=   { des-line NL }+
des-line          ::=   comp-category {HT}+ eval-name
comp-category     ::=   <name of the component category (no embedded tabs)>
HT                ::=   <horizontal-tab character>
eval-name         ::=   <evaluation module user-oriented name>
NL                ::=   <newline character>
```

Figure 32. Backus-Naur notation for damage evaluation selection files

4. User Interface

To the vulnerability analyst, MUVES exists as a central console for specifying inputs, configuring, executing, and interpreting the results of vulnerability analyses. A less visible role for MUVES, yet embedded in its design, is the task of *configuration management*. The goal of configuration management is to keep a record of all inputs and source code responsible for producing the results for a particular analysis run. This record must be preserved indefinitely, or up until such time as the user decides that the results will not be published and can be destroyed. The ambitious task of managing the disposition of files used in an analysis requires access control measures. To accomplish this, MUVES maintains a UNIX file hierarchy which is accessible through the user interface; only the MUVES administrator has permission to freely access this hierarchy. To permit higher-level logical organization of analyses and to facilitate compartmentalized access, analyses are grouped into directories called *projects*. The user can create a project and restrict multiple levels of access to specific users (access lists are described in more detail later). All input files used to run an analysis are *imported* (copied) into a project file hierarchy. They may be imported from outside the MUVES hierarchy or from another project. Output files are also kept within the MUVES hierarchy. Notably, postprocessed results, which typically fall into the category of deliverables, are not stored within the MUVES hierarchy but are owned by the analyst.

Each MUVES analysis is referred to as a *run*. A particular run may compute the outcome of one or more shots. In addition to multiple shots, multiple *threats* may be analyzed, and more than one context for damage assessment may be applied to a given run. If multiple shot specifications, threats or contexts are selected, all possible combinations of these are analyzed. A series of runs conducted while a given project is being accessed by MUVES is called a *session*. A session file is produced automatically during a session, and a run from this file can be reloaded during a subsequent session to conduct an identical or similar analysis. All diagnostics from a session's analyses are stored in a *log file*. *Final results* files are created for each run, and optionally, if an analyst needs information regarding the intermediate computations of a MUVES calculation, selected internal parameters for a particular run may be stored in an *intermediate results* file. Details about these output files will be explained later.

4.1 User Interface – The Basics

The user interface is initiated by executing the command **muves** in a UNIX shell. The user must have "$MUVES/bin" in his path for the software to work correctly. Questions concerning the appropriate path and shell variables to be set should be referred to your MUVES system administrator. When **muves** is started, the process fills the current window (or terminal screen on non-windowing systems) with a display similar to Figure 33.

The window is divided into two parts; the top portion is devoted to the hierarchical "pop up" menu system, and the bottom portion is dedicated to a scrolling-text region. In between, several status lines are reserved for prompted input, or for presenting the user with information of a transient nature such as warning messages, informative messages about on-going computations, or help messages generated by the menu system.

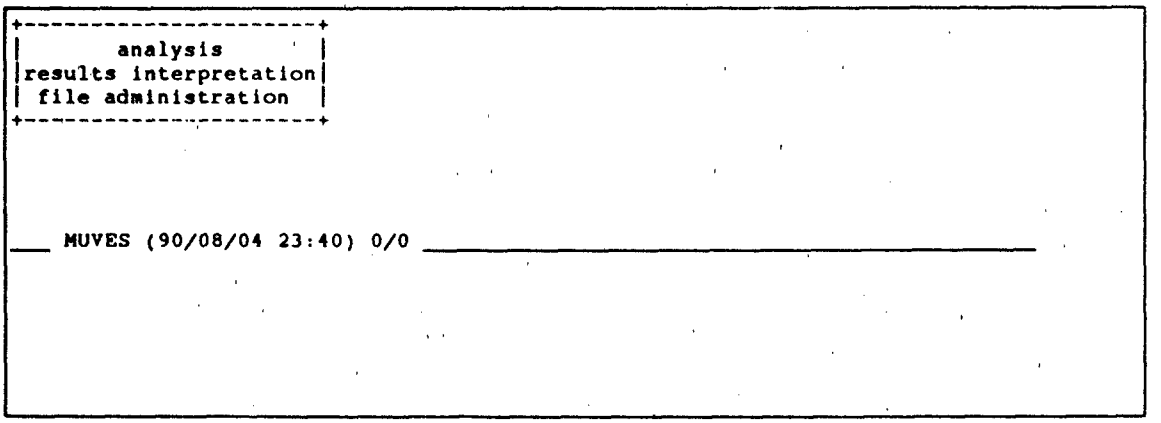


Figure 33. Terminal Window after **muves** has been Started.

The menu system hierarchy groups commands according to their function. The scrolling-text region is used to display various information such as error messages, the contents of MUVES files, the status of analyses, or the output of postprocessors. Such information is stored in an internal buffer so that the user can, with single key strokes, display different contiguous regions of that buffer.

The screen-oriented interface is terminal independent; it does not use graphics *per se* but uses the UNIX system terminal capabilities database to control the manipulation of text characters.

4.1.1 Using the Menus

MUVES menus are made-up of one or more entries. The active menu entry is always highlighted. Selection of a menu item may result in the presentation of a *submenu*. Alternatively, prompts for keyboard input may be issued, input parameters selected, or UNIX commands executed. Submenus will appear to the right of their parent as space permits, then progress downward in left to right order, overlapping other menus as necessary.

Each menu operation (*e.g.* move menu pointer, select menu item, scroll menu, close menu), is invoked by a unique command. To make the user interface easier to use, it is possible to bind these commands to keys, however, default bindings are provided. Thus, simple key strokes are used to manipulate the menus.* One key (character) must be bound to a given menu-manipulation command. The control key can be used as a key modifier. Table 4 lists the menu-manipulation commands and their default key bindings. Pressing the key indicated in the **Key** column will cause the action described in the **Command Description** column. The defaults can be overridden by each user through the use of a MUVES start-up file (see Section 4.1.4, MUVES **Start up File**). Figure 34 shows **muves** after a few entries have been selected. Note the menus are positioned left to right, with some menus overlapping those selected earlier.

* If a DMD terminal with MYX is used, a special cursor will appear. The mouse may then be used to manipulate the menus. At this time, only the DMD terminal running MYX supports mouse-driven menu manipulation.

TABLE 4. Menu Manipulation Commands and their Default Bindings.

Key	Command Name	Command Description
d	**next-menu-entry**	move menu pointer down one entry
u	**previous-menu-entry**	move menu pointer up one entry
space	**select-menu-entry**	select highlighted menu entry
h	**describe-menu-entry**	get help about highlighted menu entry
q	**quit-current-menu**	quit current menu

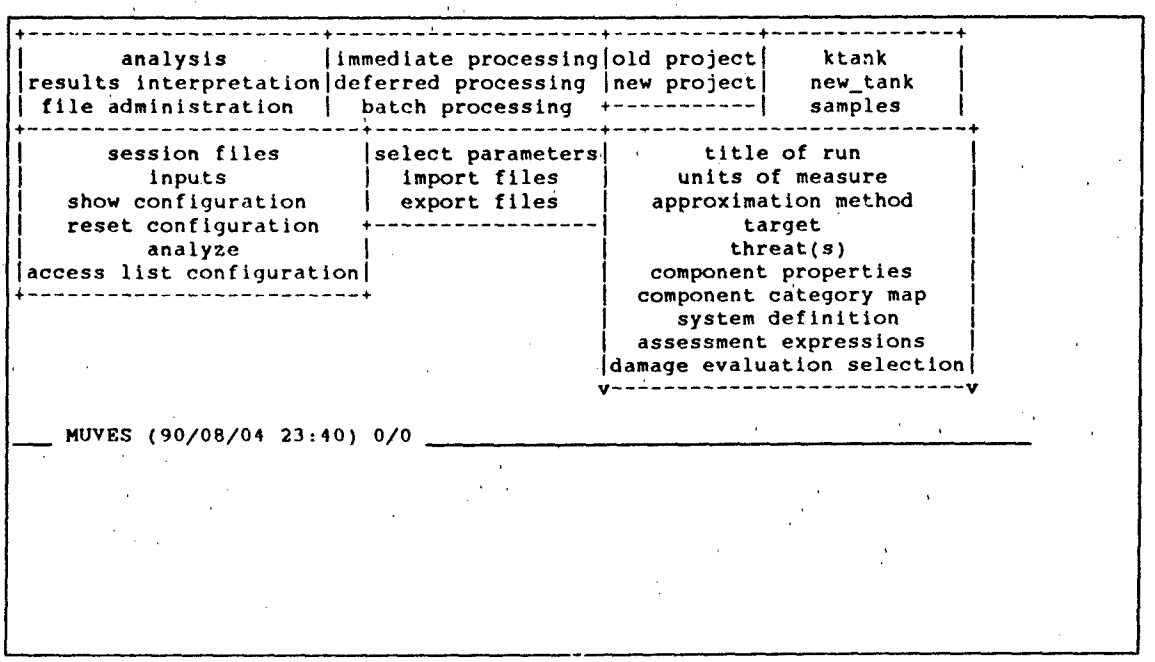


Figure 34. Scrolled Menus.

The user interface automatically determines where menus will appear based on menu size and screen size. If on a windowing terminal or workstation, the user may choose to increase the width of the window before **muves** is started to minimize menu overlapping.

Menus with many entries will be truncated to prevent them from extending outside the menu display region. The corners of a truncated menu will appear as "v" or "^". These symbols indicate that the menu must be scrolled up or down to display the other entries. Figure 34 shows a menu that has been truncated, notice the "v" in the menu's lower left and right corner. The other entries will become visible when moving the menu pointer down (using the **next-menu-entry** command) past the bottom entry (shown in the figure as **damage evaluation selection**).

Figure 35 shows a "^" in the upper left and right corners of the scrolled menu, indicating some entries have scrolled off the top. Moving back up (**previous-menu-entry**) will cause the menu to scroll down to

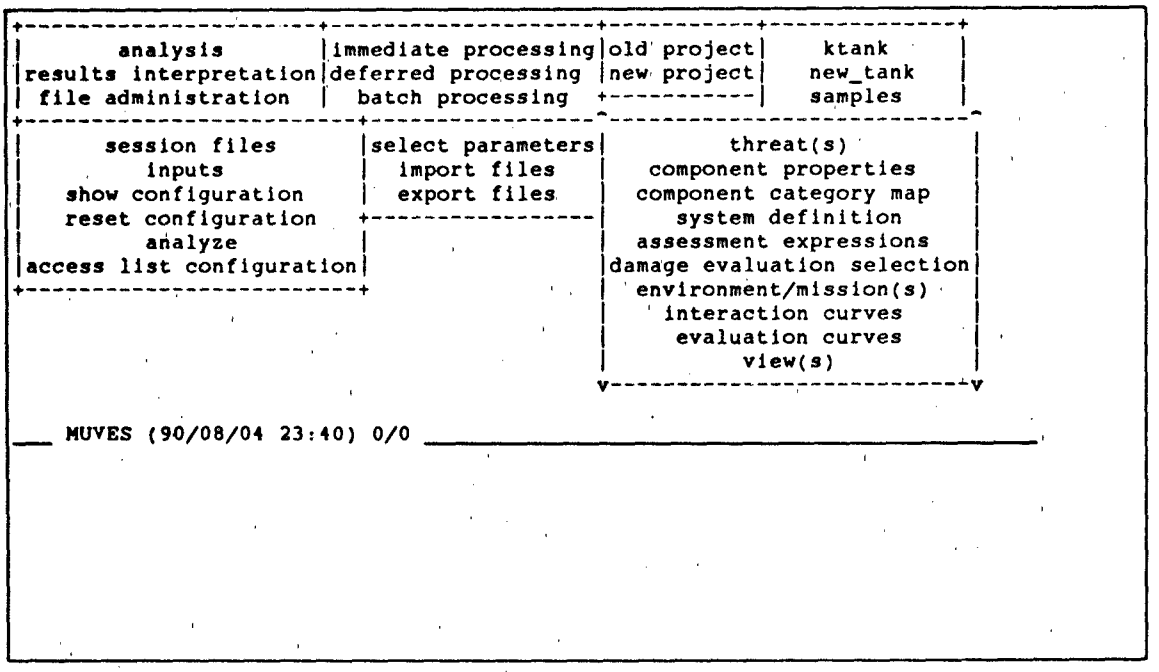


Figure 35. Scrolled Menu – Top and Bottom.

reveal these entries. When the last or first menu entry is visible, the menu corners will return to "+".†

Although all menus have a similar format, there are two distinct types – static and dynamic. A static menu always consists of the same entries. In contrast, dynamic menus are created immediately before they are displayed. Dynamic menus are necessary because they list choices whose availability is contingent upon previous events. For instance, a menu of available projects will only display the names of projects that exist in the current MUVES hierarchy for which the user has been granted access.

The menu system keeps a record of the last entry selected in a static menu; entering a static menu that has been used before will cause the menu pointer to be in the same position as it was when that menu was last exited. If a menu has never been used, the top entry will become current. Since dynamic menus can contain different entries each time they are activated, the top entry will always be current when a dynamic menu is opened.

A few menus can be accessed on demand, that is they can be opened with a command. These are called ***stand-alone*** menus because they are not found in the menu hierarchy. The commands that produce stand-alone menus are described in Table 5. The **set-up-options** command produces a menu (illustrated in Figure 36) whose first entry, **maximum menu items visible**, allows the user to change the size (number

† When running MYX on a DMD terminal, clicking the left button of the mouse will have the following effect: If the cursor is on an entry, that entry will be made current. If on the current entry, that entry will be selected. If on the top border of the current menu, the menu will scroll down if possible, and if not, the menu will exit. If on the bottom border, the menu will scroll upward if possible and beep if not. Clicking button one outside of any menu region will cause the terminal to beep.

TABLE 5. Pop-up Menu Commands and their Default Bindings.

Key	Command Name	Command Description
$	**set-up-options**	tailor user interface
@	**manage-analyses**	get status/terminate analyses
&	**enter-file-browser**	examine MUVES data hierarchy
?	**show-key-bindings**	display menu of bound keys

of lines) of the menu region being displayed. A larger menu region will allow more menus to be visible but will reduce the number of lines being displayed in the scrolling-text region.

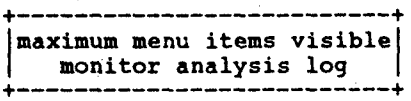


Figure 36. Setup Options Menu

This set-up menu also allows the user to monitor any logged messages from the MUVES analysis process. If the menu entry **monitor analysis log** is enabled, any analysis errors will appear in the scrolling region. Error messages are always recorded in a log file regardless of whether log monitoring is enabled.

Occasionally, the user may need to check on or terminate a running analysis. This can be done from within MUVES if the particular analysis session was initiated during the current MUVES invocation.

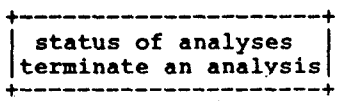


Figure 37. Analysis Management Menu

A stand-alone menu displayed with the **manage-analyses** command is provided for this purpose (see Figure 37). Selecting **status of analyses** will display information about each analysis session. If **terminate an analysis** is selected, a menu of running analysis processes is presented; the user is advised to exercise caution when choosing from this menu.

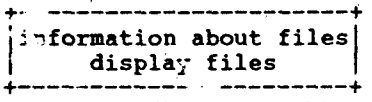
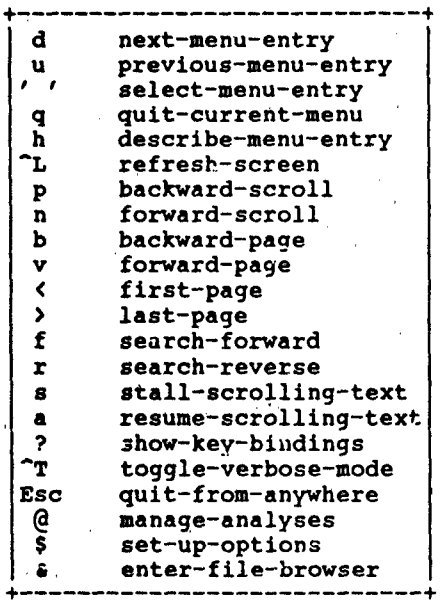


Figure 38. File Browser Menu

The *file browser* menu (Figure 38) may be entered via the **enter-file-browser** command. This menu allows users to examine files or directories under any project for which they have read access. The **information about files** entry produces listings similar to the output of the UNIX *ls*(1) command, and the **display files** entry *exports* the contents of selected files. *Exporting* is used here as a general term for sending the contents of a file somewhere; this includes providing the file as input to a UNIX command, making a copy of the file outside the MUVES file hierarchy, or displaying the file's contents in the scrolling region. MUVES owns all files within a project but an exported file copy is owned by the user.



d	next-menu-entry
u	previous-menu-entry
' '	select-menu-entry
q	quit-current-menu
h	describe-menu-entry
^L	refresh-screen
p	backward-scroll
n	forward-scroll
b	backward-page
v	forward-page
<	first-page
>	last-page
f	search-forward
r	search-reverse
s	stall-scrolling-text
a	resume-scrolling-text
?	show-key-bindings
^T	toggle-verbose-mode
Esc	quit-from-anywhere
@	manage-analyses
$	set-up-options
&	enter-file-browser

Figure 39. Key Bindings Menu

Finally, the **show-key-bindings** command produces an information menu that displays the current key/command bindings, the default bindings are shown in Figure 39.

4.1.2 Prompted Input

The menu display region is separated from the scrolling-text region by three lines: a help line, a prompt line, and an information bar. In the information bar, the name of the program, "MUVES", is displayed, followed by the date and time that the software distribution was made. The other information in the bar pertains to the scrolling region and will be discussed in the next section. The prompt line is used to solicit keyboard input from the user and to display any response that is typed in by the user. The help line is used to provide the user with additional descriptive information about a solicited response, and also to provide other information as needed. For instance, warning messages concerning user interface operation, menu help messages, or the notification of on going computations might be displayed. Information on the help line is displayed in *reverse video*, but will be illustrated using **bold face** in this document. The prompt itself consists of a request for information terminated by a default response enclosed in square brackets. When there is no appropriate default, the brackets will be empty. Examples of appropriate responses may also be included within the prompt; these would be enclosed in parentheses to the immediate left of the default response. In the example illustrated in the Figure 40, a user is being asked for confirmation before exiting **muves**.

```
Do you really wish to leave now?
Exit "muves" (y or n) [y]: n
```

Figure 40. Exit **muves** Confirmation Prompt

Appropriate responses "y or n" are enclosed in parenthesis, the default response "y" is enclosed in square brackets. In this example, the user has typed *n* which appears following the prompt. Also, note the help message which is shown in bold face above the prompt.

TABLE 6. Prompt Editing Commands

Key	Command Description
Delete	Delete the character behind the cursor.
Return	Enter response.
^A	Move the cursor to the beginning of the line.
^B	Move the cursor backward one character.
^D	Delete the character under the cursor.
^E	Move the cursor to the end of the line.
^F	Move the cursor forward one character.
^G	Abort the prompt without submitting a response.
^K	Erase all characters (kill) from the cursor to the end of the line.
^P	Insert the default response at the cursor location.
^R	Redraw the line.
^U	Erase all characters to the left of the cursor.
^V	Escape (insert) the next character typed.
^Y	Insert the contents of the kill buffer (see ^K) at the cursor.

While typing a response, some line-editing commands are available to the user; these are described in Table 6. With the exception of the first two keys, "Delete" and "Return", control keys are used as indicated by the "^" in addition to the letter. Use of upper case letters to describe control keys is a popular convention, but does not denote the use of the "Shift" key; only the "Control" key must be depressed while striking the letter key. Text is inserted at the cursor position. Using these commands to move the cursor allows the user to correct typing errors or modify previous responses. For instance, the "^P" command will insert the default response at the cursor position. In this way, a previously entered response can often be edited to produce something similar.

4.1.3 Scrolling the Text Region

The scrolling-text region occupies the lower portion of the display and presents various information communicated to the user by MUVES. Lines of text are inserted at the bottom of this region and existing lines move upward to make room for newer ones. A finite number of lines of text will fit in this region; this number depends on how many lines can be displayed on the user's terminal or window and how large an area he has allocated for the menu region. Once this region fills with text, the oldest lines will cease to be displayed in favor of newer ones. Various commands are available to the user to enable him to scroll backwards or forwards to display older or newer lines of text respectively. Rather than scroll one line at a

time, the user may advance the text by the number of lines that can be displayed at one time; this is referred to here as a *page* of text. The user also has the option of advancing to an arbitrary line of text, such as the oldest or newest that is available; for the purpose of this discussion, this type of operation is called *jumping*. Finally, the user may jump to a page containing a *regular expression*** ; this is called *searching*. A finite number of lines are stored in an internal text *buffer* for this purpose; this number can be set in the **muves** start up file (see Section 4.1.4, **MUVES Start up File**). Once the text buffer is full, the oldest lines will be discarded first. One thousand lines are saved by default.

There are many commands available to control the scrolling of text. These commands are bound to keys, just like the menu-manipulation or prompt-editing commands.

TABLE 7. Scrolling Region Commands and their Default Bindings.

Key	Command Name	Command Description
p	**backward-scroll**	move text window down one line
n	**forward-scroll**	move text window up one line
b	**backward-page**	move text window down one page
v	**forward-page**	move text window up one page
<	**first-page**	jump to first page in text buffer
>	**last-page**	jump to last page in text buffer
f	**search-forward**	look for next occurrence of regular expression
r	**search-reverse**	look for previous occurrence of regular expression
s	**stall-scrolling-text**	halt display of buffered text
a	**resume-scrolling-text**	resume display of buffered text
^L	**refresh-screen**	redraw the MUVES display

Table 7 lists the available text buffer commands and their default key bindings. The first six commands allow the user to display text saved in the buffer. The information bar, above the text display region, indicates the range of lines currently visible to the user and the total number of lines in the text buffer. The next two commands can search forward or backward in the text buffer for any given regular expression. The two commands after that allow the user to stall or resume scrolling in the text region. The information bar indicates whether the text region is stalled; by default it will be stalled initially. When the text region is stalled, information still flows to the text buffer, but is not displayed on the screen. Any command given by the user which changes which lines of text are being displayed will cause the text region to revert to the stalled mode. When not stalled, the text region scrolls to display information as it becomes available. Figure 41 illustrates the scrolling window where lines 1-18 are

** A regular expression is a notation for specifying and matching strings. See the ed(1) UNIX manual entry, or *The AWK Programming Language* by Aho, Kernighan, and Weinberger.

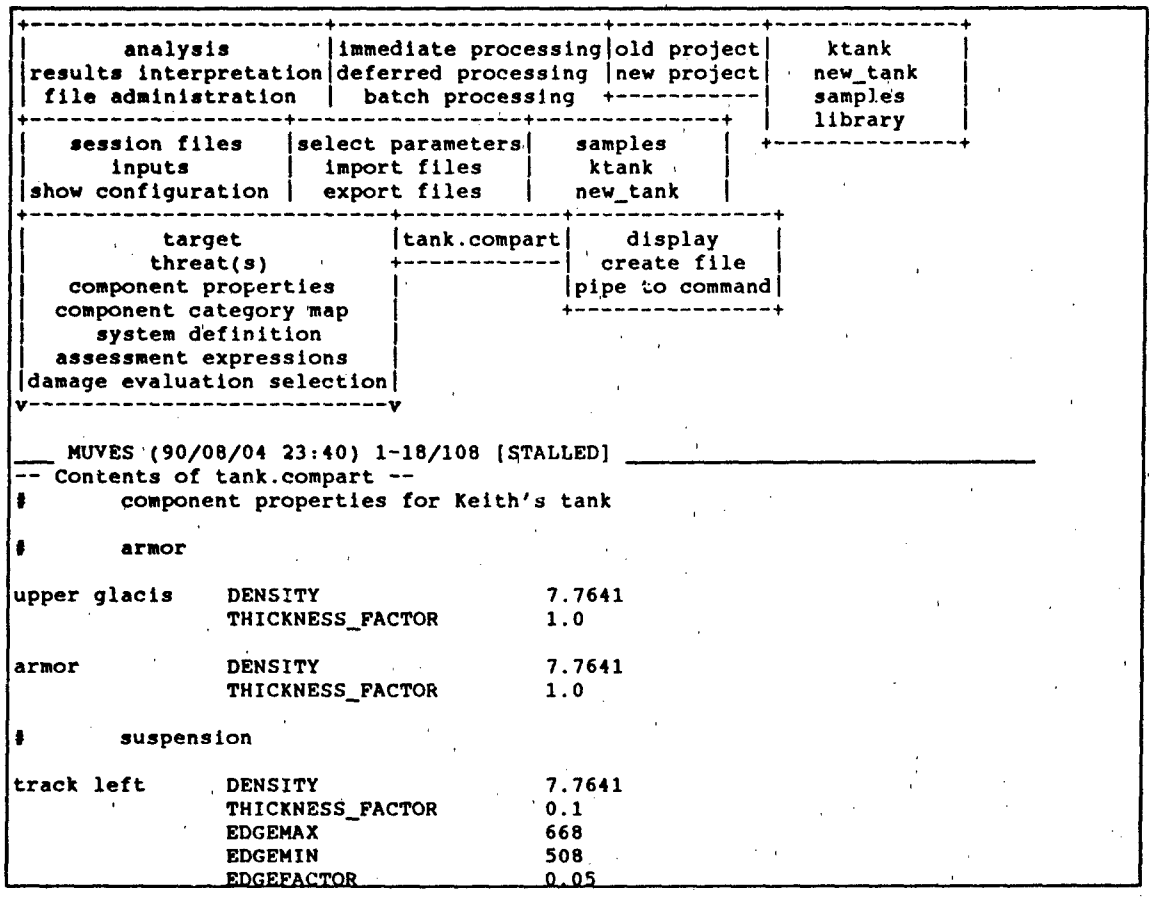


Figure 41. Scrolling Region

displayed. The information bar indicates that there are 108 lines in the text buffer, and the display window is stalled. It is recommended that the display be stalled prior to displaying a large file because it could take a long time to scroll through the entire file. On typical hardware platforms, information can be sent to the screen faster than it can be displayed. On a fast computer, volumes of text can be sent in a split second. Meanwhile, this information is stored in the display hardware's buffer and/or in the operating system's kernel buffer. In either case, once this buffering occurs, the MUVES user interface cannot stall the display until all of the buffered information has been displayed. However, if the display has been stalled in advance, the user can control which lines of text are displayed via the commands outlined in Table 7. Rather then scrolling the text to the end of the buffer, the user may want to "jump to the last text page in buffer" with the **last-page** command before scrolling is resumed. The final command in Table 7, **refresh-screen**, redraws the entire MUVES display.

4.1.4 MUVES Start up File

Since users will most likely want to modify **muves**' default behavior, the ability to read a "start up" file has been provided. Whenever **muves** is started, if the file **.muvesrc** exists in the user's home directory then it is read by the user interface. This file contains directives to configure various items. A directive is a "keyword" recognized by the user interface which sets certain parameters.

```
# My favorite key bindings.
BindKeyToFunction       ^N      next-menu-entry
BindKeyToFunction       ^P      previous-menu-entry
BindKeyToFunction       /       search-forward
BindKeyToFunction       Del     quit-from-anywhere

# Set maximum number of menu items visible.
MenuItemsVisible        15

# Number of lines to store in the text buffer.
TextLinesSaved          2048

# Monitoring of analysis log disabled.
MonitorAnalysisLog      false

# Defaults for colorsil (color silhouette) postprocessor options.
BinCellLocations        true
CellDisplaySize         20 20
CenterAtTargetOrigin    true
ExplodedViewFactor      1.0
FrameBufferDevice       myhost:/dev/sgipl
ImageDimensions         768 768
InterpolateColors       true
MonochromeDisplay       false   # default
ShowColorKey            true
ShowGridPlaneAxes       true
SpaceBetweenCells       2
TargetSpaceCellSize     50.0 50.0
```

Figure 42. Sample ".muvesrc" File

A sample .muvesrc file appears in Figure 42 that shows all supported directives. Just like all other MUVES files, blank lines and comment lines, *i.e.*, lines starting with a '#' character, are ignored by MUVES. Any number of tabs or spaces are allowed between a directive and its argument(s). Directives can be in any order.

The first directive in this sample file is **BindKeyToFunction**. This directive allows the user to bind keys on a keyboard to any of the aforementioned command names (Tables 4, 5 and 7). Valid keys are any "ordinary" key on the keyboard, *i.e.*, letters, numbers, or symbols. There are also three special "words" that are recognized as keys. The escape key is represented by "`Esc`", "`Del`" represents the delete key, and "`Tab`" indicates the tab key. To provide even more binding possibilities, the control key, denoted by the circumflex symbol ('^'), may be used in conjunction with other keys (*i.e.* the range 'A' to 'Z', '@', '[', '\', ']', '^', or '_') as a modifier. Based on the key bindings of Figure 42, the **next-menu-entry** command is issued by holding down the control key and then pressing the 'N' key. This will move a menu

pointer down to the next menu entry.

The number of menu items visible in the upper region of the **muves** display is configurable at start up by using the **MenuItemsVisible** directive and the number of lines of text saved for the scrolling-text region is configurable with the **TextLinesSaved** directive. Both of these commands take an integer argument. The next directive in Figure 42, **MonitorAnalysisLog**, allows the user to set log file monitoring to `true` or `false`. The last block of directives refer to options whose usage is described in the "colorsil" postprocessor documentation (see Section 5.2.2, **Postprocessors**, **Color Silhouettes**). These options are also configurable from the postprocessor menus (see Section 4.2.2, **Results Interpretation**).

TABLE 8. Start up File Directives

Directive	Arguments
BindKeyToFunction	ordinary key ^ followed by A-Z@[]^_ Esc Tab Del ' ' command-name
MenuItemsVisible	integer
TextLinesSaved	integer
MonitorAnalysisLog	true/false
BinCellLocations	true/false
CellDisplaySize	integer integer
CenterAtTargetOrigin	true/false
ExplodedViewFactor	real
FrameBufferDevice	string
ImageDimensions	integer integer
InterpolateColors	true/false
MonochromeDisplay	true/false
ShowColorKey	true/false
ShowGridPlaneAxes	true/false
SpaceBetweenCells	integer
TargetSpaceCellSize	real real

The **muves** start up file directives are summarized in Table 8.

4.2 The Menu Hierarchy

The following sections document each **muves** menu and explain their entries in detail. First, the main menu is described, followed by descriptions of the submenus below each of the three main menu entries.

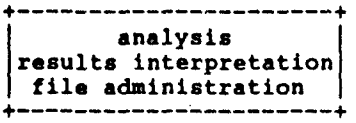


Figure 43. Main Menu

In the main menu, Figure 43, the user must choose between three main tasks: configure/execute analyses, examine analysis results, or administer analysis input/output files.

4.2.1 Analysis

To run an analysis, input files must be set up and analysis parameters must be specified. All input selection and analysis configuration as well as the actual analysis execution is done through the **analysis** entry (Figure 43). Thus selecting **analysis** will open submenus that configure a MUVES *session*.

MUVES run time varies with the complexity of the target and the type of analysis. The submenu in Figure 44 shows the three available processing alternates. If **immediate processing** is selected, analysis runs are executed in sequence as directed by the user. While a given analysis run is executing, the user may configure other runs and queue them up as part of the current session. The user is expected to wait until all runs in the session complete before leaving the current project (see below) or exiting **muves**.

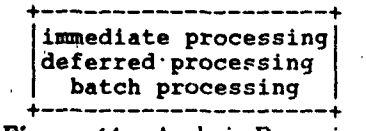


Figure 44. Analysis Processing

Many sessions can be set up and executed, one after the other, with **deferred processing**. This mode starts each session as a background process, eliminating the wait for a previous run or session to complete. Some UNIX systems offer a *batch* queue. The **batch processing** option accesses whatever queueing system is configured on your particular UNIX system and queues each MUVES session. Batch queueing is desirable for long sessions or when the user wants to configure many sessions at once. The queueing system will control the processes, running them as machine resources become available and possibly restarting processes after machine shutdowns. Queueing may not be available on every system. Consult your system's administrator for more information.

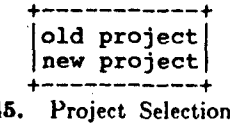


Figure 45. Project Selection Submenu

For reasons mentioned previously, the interface organizes its file hierarchy by *projects*. The analysis menu allows creation of new projects or selection of old (existing) projects (see Figure 45). The **old project** selection will display a menu of available projects. For instance, the "samples" project comes with the MUVES software and is intended to provide examples of input and output files. There may also be a "library" project set up by your MUVES administrator to provide useful versions of input files that may require little or no change. Finally, any other projects created by MUVES users on your particular host computer for which you have write access will be listed (see Section 4.2.1.4, **Access List Configuration**, for more information). This allows projects to be shared by others. When **new project** is selected, the user interface prompts for the name of the new project. It is suggested that meaningful names be used for projects.

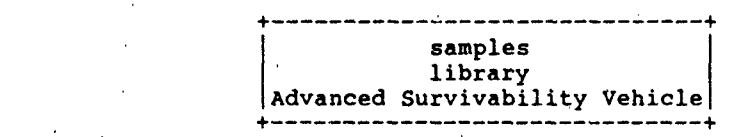


Figure 46. List of Projects Submenu

As an example, Figure 46 shows a menu of available projects. The project "Advanced Survivability Vehicle" has been created. Later, other members working on the "Advanced Survivability Vehicle" study can easily select this project to run their analyses.

Each time an old project is selected or a new project is created, a "MUVES session" is started. All analysis configuration parameters, including the names of input and output files, are recorded during a session in a session file. MUVES sessions are numbered in sequence for a given project, beginning with 1. Each time an analysis is executed a MUVES run is recorded. Therefore, a session contains one or more runs. Run numbers consist of the session number, followed by a decimal point and sequence number for that session, beginning with zero. For example, if session 3 contains 2 runs they are numbered 3.0 and 3.1. Each run within a session must be assigned a title before it can be executed; the title of the first run becomes the session's title. Although a session or run stored in the MUVES file hierarchy can be referenced by the session or run number, it is recommended that a meaningful title be provided. Various selection menus will use the number and title to reference a particular session or run. The title can aid the analyst in consistently identifying a run and its associated intermediate and final results files.

Session files permit an analyst to refer back to previous runs to compare results or to set up a new run with similar parameters. Archiving them with the corresponding results and requisite input files produces a record of the analysis for future reference. If the analyst wishes to run a variation on a previous analysis, he can load the corresponding session file and change the relevant parameters via the user interface (See Section 4.2.1.1, **Analysis Configuration – Sessions**).

Once the type of processing is selected and a project is chosen, necessary input files and configuration parameters must be selected. The analysis configuration menu, shown in Figure 47, is broken into six selections. The submenus of each selection are described in the sections that follow. Note, the current session continues until the analysis configuration menu is terminated. This will require confirmation by the user; he will be prompted as shown in Figure 48. The user can end the session by typing "y", or continue the **same** session by typing "n".

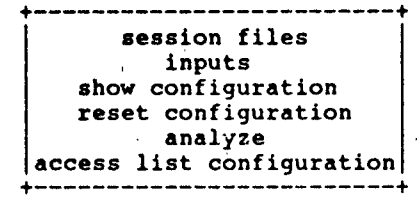


Figure 47. Analysis Configuration Submenu

```
Quitting this menu will complete the current session.
End the analysis session now (y or n) [y]:
```

Figure 48. End Session Confirmation Prompt

4.2.1.1 Analysis Configuration – Sessions

The **session files** entry in Figure 47 gives the analyst access to previously recorded sessions. Selecting this entry produces the submenu illustrated in Figure 49.

Figure 49. Session Files Submenu

Selecting the **load session file** entry will display a menu of session files available in the current project.

```
Jul  5 90    7 junk
Jul  5 90    6 test multiple runs
Jul  5 90    5 test run
Jul  2 90    4 cellxcell
Jul  2 90    3 First run, test input files
Jul  2 90    2 cellxcell test run - take 3
Jun  1 90    1 cellxcell test run
```

Figure 50. Load Session File Submenu

For instance, Figure 50 shows seven available sessions listed in reverse-chronological order, each one denoted by the date it was created (fields 1-3), the session number (field 4), and the session title (comprises the rest of the entry). Note that the session title is really the title of the first run in that session. Once a session file entry is selected, a submenu will appear and list the available runs in chronological order.

```
3.0 First run, test input files
3.1 Modified properties file
3.2 3rd run - its got to be right
```

Figure 51. Run Menu

Each run is denoted by its run number and title (see Figure 51) which correspond to the MUVES results file number and title respectively.

```
Reset queues first (y or n) [n]:
```

Figure 52. Reset Configuration Prompt

After selecting one of the listed runs, the user will be prompted as illustrated in Figure 52. Typing "y" will replace the current analysis configuration with the configuration found in the selected MUVES run. Alternatively, if the user types "n", the configuration found in the selected MUVES run will be merged with the current configuration.

The last entry in Figure 49 is **export**. As previously noted, all session files, input files, and results files are stored in the MUVES file hierarchy. The *export* facility allows these files to be copied (exported) outside the MUVES file hierarchy; once exported, the files are owned by the user. Once **export** is selected, a list of the current project's session files will appear like those in Figure 50. After selecting a session file, the export submenu will appear as in Figure 53.

```
+----------------+
|    display     |
|  create file   |
|pipe to command |
+----------------+
```

Figure 53. Export Submenu

A file can be exported to the text buffer by selecting the **display** entry. The contents of the selected file can then be viewed by the user in the scrolling-text region. A new file, outside of the MUVES file hierarchy, can be created with the **create file** entry. In this case, the interface will prompt the user with **Copy to []:**, and the user may enter a file name. By default, the new file is created in the *current directory* (whatever directory was current at the time the user started up the user interface). Full paths can be given if the user wishes to place files in other directories. A "~" may precede a path or file name. In this case, the user interface converts "~" to the user's home directory, based on the value of the $HOME shell environment variable. Thus, **Copy to []:** *~/new* creates a file called *new* in the user's home directory, **Copy to []:** */usr/tmp/new* creates a file called *new* under the directory */usr/tmp*, and **Copy to []:** *new* creates a file called *new* in the user's current directory. If a file named *new* already exists, it will be overwritten.

The export submenu also allows the user to provide the selected file as input to any UNIX command that reads its standard input. For instance, a session file can be sent to a printer by selecting **pipe to command**. The interface will then prompt for the command. For example, here the user has issued the *lpr* command: **Command []:** *lpr*. Note that the UNIX pipe symbol, "|", may be embedded in a command. This enables the file to be formatted by *pr* before being sent to *lpr* as follows: **Command []:** *pr | lpr*. Any output to standard out or standard error from an entered command will be appended to the text buffer. The **pipe to command** entry can be further exploited by using commands such as **Command []:** *grep target*. This will display all lines of the selected file containing the word *target*. The user has all the above means with which to view various session files to assist in a file selection.

Once a session file has been selected and loaded, the analyst may make incremental changes to the configuration of the current session as desired (see Section 4.2.1.2, **Analysis Configuration – Input Selection**, below). This provides a quick means for the analyst to do comparisons. As an example, a run could be loaded from a previous MUVES session in which target "A" was analyzed. Using the input parameters submenu, the target selection could then be altered to target "B", with the only other change in inputs being the title of the run. The "reusing" of a previous session file in this example assures **all** input parameters are identical except the target, providing an efficient means of conducting a controlled target comparison (*n.b.*, if any randomization such as grid cell location has been selected, a seed must be specified to insure that identical shots are computed).

4.2.1.2 Analysis Configuration – Input Selection

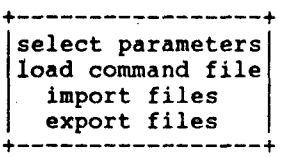


Figure 54. Inputs Submenu

The user selects inputs and configures the analysis with the **inputs** menu entry in Figure 47. The inputs submenu in Figure 54 allows the analyst to select various configuration parameters, to read commands from a file, and to copy input files into and out of the MUVES file hierarchy.

4.2.1.2.1 Input File and Input Parameter Selection

Choosing **select parameters** from the menu in Figure 54 allows the user to configure the analysis. As previously noted, a session file could have been loaded or the user could have loaded a command file (see Section 4.2.1.2.2, **Loading a Command File**). If so, the user may only need to alter a few selections. If previous session files are not desirable, the user can select the necessary parameters directly.

Each of the entries displayed in Figure 56 is used to select a parameter or input file to configure the current analysis run. As each selection is made, the interface sends a description of the selection to the scrolling text region. Many selections display submenus of available input files under the current project in the MUVES file hierarchy. Since the user can only select input files that exist in the MUVES file hierarchy, a means is provided to copy in or *import* data files/directories into the project directories, see Section 4.2.1.2.3, **Import – Moving Input Files into the MUVES File Hierarchy**, for an explanation of the importing facility.

A means for adding comments to the current session file is provided by the first entry, **session comment(s)**, shown in Figure 49. The menu entries that are plural, *i.e.*, ending in "**(s)**", allow for multiple selections to be *queued*. Selecting a plural entry from the menu in Figure 53 will always produce the queue menu in Figure 56. A description of the contents of the selected queue is displayed in the scrolling text region by selecting **show queue**. The members of the queue will be numbered and listed in the order that they are to be processed. To add comments to the session file, the **append to queue** entry

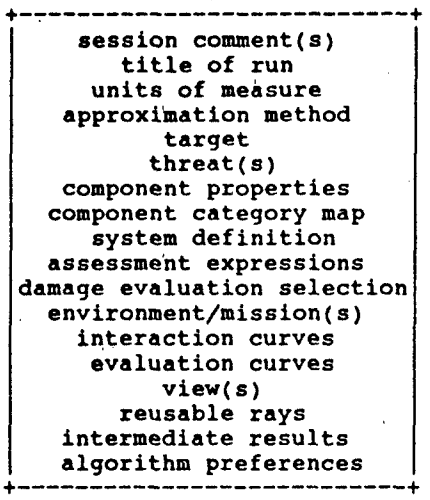


Figure 55. Input Parameter Selection Menu

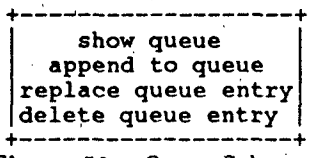


Figure 56. Queue Submenu

is used. When selected, this entry will prompt the user for a one-line comment (up to 254 characters). The **append to queue** menu entry may be selected repeatedly to insert blocks of text. After one or more comment lines have been added to the queue, it is possible to delete or replace an entry in the queue. Selecting **replace queue entry** will first produce a menu listing the currently queued items, from which the entry to be replaced must be selected. The **delete queue entry** menu lists the currently queued items. Selecting one of the items in the list will delete it from the queue. When an analysis is submitted, all comments are recorded at the top of the session file in the order that they appear in the queue.

The next entry in Figure 56, **title of run**, is used to assign a title to the current run. As previously noted, the title is mandatory and is recorded in both the session and final results file. The next entry in the menu is for selecting units of measure for interpreting certain inputs entered via the user interface when describing viewing parameters (discussed later in this section). For instance, if the user wishes to express all coordinates in inches, he should select "inches" from the **units of measure** submenu. Only one choice of units can be made per analysis run; it is unimportant if the units are selected before or after the numbers are input.

As described in the **System Design** section, MUVES is intended to eventually support many approximation methods. A submenu of available methods are displayed by using the **approximation method** entry. The desired method to use for an analysis run must be selected from this submenu.

There are many input files that must be specified before an analysis is run. Each input file/directory is described in the **Analyst Supplied Input Files** section of this document. First, the user must select a target and one or more threats. The **target** menu entry provides the list of the current project's target

directories, and the **threat(s)** entry provides the list of the current project's threat directories. The target is selected by choosing an entry from the provided list. Only one target may be specified for a run. However, MUVES provides the ability to investigate more than one threat at a time. If multiple threats are specified, MUVES will iterate through these during the analysis run. A queue of threats is specified in a manner similar to the queueing of session comments as described above; the only difference is that when the **append to queue** entry is selected, rather than prompting for the entry, a submenu listing the available selections is presented. Each time an entry is selected from this menu, the selection is added to the queue. The queueing mechanism allows the analyst to specify multiple threats, and/or different ranges of a threat to be played against one target. Also multiple views and environment/mission pairs may be queued up and applied in all combinations to other input selections.

The next five menu entries of Figure 55, **component properties, component category map, system definition, assessment expressions,** and **damage evaluation selection**, are used to select input files. Information about the input files for each entry can be found in the **Analyst Supplied Input Files** documentation section: **Component Properties File, Component Category Map File, System Definition File, Damage Assessment Expression File,** and **Damage Evaluation Selection File**, respectively.

The user must select one or more *contexts* for the evaluation of a selected target's functionality. A context is described in the form of an environment/mission pair which is defined in the damage assessment expression file. The **environment/mission(s)** submenu is generated dynamically using information found in the damage assessment expression file. Therefore, the **assessment expressions** entry must be used before **environment/mission(s)**. Also note that if the damage assessment expression file is changed or its contents modified, environment/mission pairs previously queued may no longer be valid.

```
+------------------------+
|   typical:firepower    |
|   typical:mobility     |
| typical:catastrophic   |
+------------------------+
```

Figure 57. Assessment Expression Environment/Mission Menu

Figure 57 is an example of an environment/mission pairs menu. A queue of environment/missions pairs may be constructed by selecting entries from this menu.

The next two entries in Figure 55, **interaction curves** and **evaluation curves**, allow the selection of data files to be used for target/threat interaction and damage evaluation, respectively. The contents and usage of the interaction and evaluation curves varies with approximation method (consult the appendix to this report which deals with the selected approximation method). Additionally, the general format of these files is outlined in the **Analyst Supplied Input Files** section of this document under **Interaction Modules Inputs** and **Evaluation Modules Inputs**.

The user must specify at least one *view* for an analysis. A view, in the general sense, defines an initial location and direction from which a specified threat or threat group is launched at the specified target. Selecting **view(s)** from the "Input Parameter Selection Submenu", Figure 55, allows the user to queue up the desired views (see the "Queue Submenu" in Figure 56).

When appending views to the queue, the user has two choices: a view input file can be selected or the user may choose to define a view using the user interface.

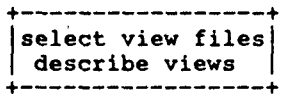


Figure 58. View Queue Submenu

Choosing the **select view file** entry, Figure 58, will produce a menu of the view files available for selection in the current project. View input files contain predefined descriptions; the syntax is detailed in the **Analyst Supplied Input Files** section under the heading **View File**.

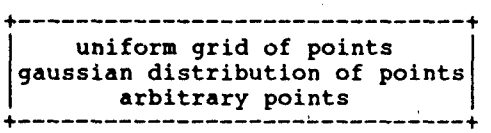


Figure 59. View Pattern Submenu

Selecting **describe views** will produce the menu in Figure 59 which groups the view description into three types. The **uniform grid of points** entry allows the specification of an evenly spaced two-dimensional array of cells (cell spacing may be different in the horizontal as compared to the vertical direction) from which shots will originate. The user is prompted for a firing direction in terms of azimuth and elevation, origin of shot (random or center of cell), cell spacing, grid offset, and the extent of the grid (or full view). The latter three grid attributes have distinct horizontal and vertical components. If random shot origins are selected, the analyst is given the opportunity to specify a seed for the pseudo-random number generator. Thus, by specifying a seed from a previous run, an identical grid of shots can be produced. If no seed is specified, the pseudo-random number generator will commence where it left off on a previous grid; a default seed value of one is used initially if none has been provided by the user. The random number generator seed used for a grid is logged in the final results file. If the user has specified a seed, it will also appear in the session file. A bivariate normal distribution of hit locations can be specified with the **gaussian distribution of points** menu entry. This specification includes azimuth, elevation, number of points in the distribution, horizontal and vertical standard deviation, and pattern offset. Individual shots can be described using the **arbitrary points** entry.

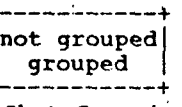


Figure 60. Shot Grouping Submenu

When the **arbitrary points** entry is selected, the "Shot Grouping Submenu" illustrated in Figure 60 will appear. The user has the option to include one discrete shot per view, or a number of shots can be grouped in a view. If grouped shots are described, the user is prompted to name the group. It is important when choosing views for postprocessing that these names are unique and descriptive in some way. Note that only one view at a time can be postprocessed; therefore, a grouping of shots will correspond to the eventual combined interpretation of their results.

Figure 61. Shot Coordinate Menu

Each shot is described either as two-dimensional grid-plane coordinates or as three-dimensional target-space coordinates. This option is selected with the menu illustrated in Figure 61. In either case, viewing direction is expressed with azimuth and elevation angles. Both methods express coordinates in the units of measure selected from the user interface and describe a point through which the shot will pass. The grid-plane coordinates refer to a point in the plane of the view using horizontal and vertical offsets from the origin of the target description; the vertical direction is aligned with the Z axis unless a top view has been selected and then it is aligned with the X axis. Target-space coordinates represent a point in the target-description's coordinate system. After selecting the **grid-plane coordinates** menu entry, the interface will prompt for the azimuth, elevation and shot location in horizontal and vertical view-plane coordinates. If **target-description coordinates** is selected, the interface will prompt for the azimuth, elevation and shot location in X, Y, Z, target space. More can be learned about shot specifications by using the help messages provided with the various menus and by consulting the **Analyst Supplied Input Files** section under **View Files**.

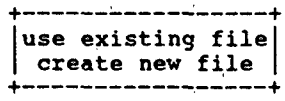


Figure 62. Setting Reusable Rays Submenu

When making a MUVES run, analysts have the ability to capture all ray traces in binary format to a file that may be used instead of actual ray tracing for subsequent analyses. Selecting **reusable rays** from the menu depicted in Figure 55 displays the submenu illustrated in Figure 62. To enable the reusable ray feature, either **use existing file** or **create new file** may be selected. If the latter is chosen, and the file exists, the user will be prompted for confirmation before proceeding to overwrite the existing file. Reusable rays are further explained in the **Analyst Supplied Input Files** section under the heading **Reusable Ray File**.

As mentioned before, each MUVES run creates a final results file containing the output from the configured analysis. If the analyst requires information computed during an analysis, **intermediate results** should be selected from the menu in Figure 55. Once this entry is chosen, MUVES will store all geometric, component, and threat information for the current analysis in an intermediate results file. This file is stored in the MUVES data hierarchy and labeled with the run number. Although these files are written in binary format to conserve space, they can become quite large; therefore, analysts should create them only when necessary and remove them as soon as possible. The primary use for intermediate results files is to give the analyst information regarding internal computations. One use for this information is to help pinpoint problems with a target or threat description when anomalies are discovered in the final results. See Section 4.2.2, **Results Interpretation**, for more information on the intermediate results files.

Certain approximation methods provide the analyst with the opportunity to choose between two or more algorithms for certain calculations. For details about the algorithm choices for a particular approximation method, consult the documentation in the appendix pertaining to that method. Selecting the **algorithm preferences** entry in Figure 55 will display the available algorithm choices for the previously selected approximation method. Since more than one preference can be chosen for an analysis, the selections are stored in a queue. This queue is manipulated in the same manner as other queued entries described earlier in this section.

4.2.1.2.2 Loading a Command File

Command files are identical in syntax to session files‡ (a session file is a command file generated by a **muves** session). They can be exported session files that the user edited or could conceivably be generated by some other program or UNIX shell script. This menu entry can only be used to load files from outside of the MUVES file hierarchy; if the user wishes to load a session file from another project, he can export that file first, edit it if necessary, then load it as a command file. This particular facility is provided for the user who has a requirement to run such a large number of analyses that use of the menu system to select individual inputs is prohibitive.

Once the user has selected the **load command file** entry from the menu depicted in Figure 54, he will be prompted for the path name of a file. The same rules apply to the naming of a file as described above for exporting session files (see Section 4.2.1.1, **Analysis Configuration – Sessions**); the file must reside outside of the MUVES file hierarchy. Next, the user will be given the choice as to whether or not to execute the entire file. If he chooses affirmatively, the entire contents of the file will be processed (each run will be submitted). Otherwise, a menu of runs found in the specified file will be presented similar to the one in Figure 51. The remainder of the procedure is identical to the loading of session files; the analysis will not be submitted, only the current configuration will be updated. The user then has the choice of modifying the selections before submitting the analysis.

4.2.1.2.3 Import – Moving Input Files into the MUVES File Hierarchy

Since only the files from the current project can be selected for an analysis, the analyst must *import* (copy) needed files to the current project. Importing is accomplished by selecting the **inputs** menu item from the "Analysis Configuration Submenu" (Figure 47) followed by **import files** from the "Inputs Submenu" (Figure 54). Most types of input files are do not depend on other files; the exceptions are *target* and *threat* files. A directory of several files are required to describe a target, and the entire directory is referred to as the target description. The same goes for threats. Target and threat directories can be imported, in whole or in part, by specifying a directory or a file name within a directory. When importing

‡ The syntax of the command files is not described in user-level documentation and the editing of these files is not a recommended practice except for the expert user.

files or directories, the analyst has two choices for places to copy them from: an explicit full path name or another project directory.

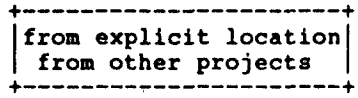


Figure 63. Import Files Submenu

Thus, the menu in Figure 63 would be shown when the **import files** entry is chosen. If **from explicit location** is selected, a list of all possible input file and directory types to import is displayed as shown in Figure 64. The type of file or directory being imported must be selected in order for the user interface to copy it to the proper location in the MUVES data hierarchy.

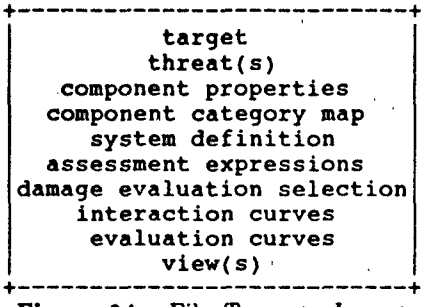


Figure 64. File Types to Import

If type selected pertains to a single file, the user will be prompted for the path name of the file to be imported. Otherwise, if the type of input is a directory (*i.e.* **target** or **threat(s)** was selected), the menu in Figure 65 will appear.

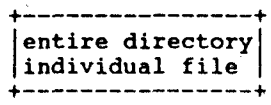


Figure 65. Entire Directory or Individual File Menu

If the user selects **entire directory**, he will be prompted for the path name of the directory to be imported. Otherwise, if he selects **individual file**, he will be provided with a menu of available destination directories of the appropriate input type. Upon selecting a directory, the user will be prompted for the path name of the file to be imported. The same rules apply to the naming of a file/directory or path as described above for exporting session files (see Section 4.2.1.1, **Analysis Configuration – Sessions**). Once the name of the file or directory to be imported has been entered by the user, he will be prompted for a name to give the new file or directory.

Enter path name to file being imported.
Import: *ktank*

Figure 66. Import File Prompt

For instance, if **component properties** was selected, the prompt would appear as in Figure 66; the user has typed in the name *ktank*. Next, the analyst is asked to enter the name of the file.

```
Enter name of new file for this project.
Copy to [ktank]:
```

Figure 67. New File Prompt

Continuing the same example, the prompt would appear as in Figure 67. The default name, taken from that of the file being copied, appears in the "[]".

When the user is prompted for the name of a file being imported, and he enters the name of an existing file, the following safeguards exist to prevent unwanted file overwriting:

- If the existing file is currently being used in an analysis, the user will be informed and the import will not be permitted.
- Similarly, if the existing file was used in a successful analysis, but the run is not yet archived, the import will not be allowed.
- Otherwise, the user will be asked for confirmation before overwriting the existing file.

See Section 4.2.3, **File Administration**, for more information on archival of input files.

If the analyst wishes to get a copy of a file/directory from another project directory then the **from other projects** entry is selected from the menu depicted in Figure 63. Next, a project is selected from a menu of available projects, and the file type is selected from the "Files Types to Import" menu (Figure 64). If **target** or **threat(s)** is chosen, a menu of available directories is presented. Once the user selects a directory, a menu identical to the one in Figure 65 is displayed. If the user chooses to import the **entire directory**, he will be prompted for a name to give the new copy; the original name will be used as a default. *The same rules apply to overwriting existing directories as outlined above for files.* Otherwise, if **individual file** is selected, a menu of available files will be presented. Once the user selects a file to import, he will be prompted for the name of an existing directory (of the appropriate type) in which to place the new file, and finally he will be prompted for a name to give the file. If the input type selected from the menu in Figure 64 was not a target or threat, then it must pertain to a stand-alone input file. In this case, a menu of available files of this type is presented. Upon selecting a file from this menu, the user will be prompted for a name to give the new file.

4.2.1.2.4 Export – Moving Input Files out of the MUVES File Hierarchy

Input files/directories that exist in the projects can be *exported* (copied) outside the MUVES hierarchy with the **export files** entry of the **inputs** menu entry in Figure 47. A menu of projects for which the user has read access will be presented. Once a project has been selected, a menu identical to the one in Figure 64 is presented. If a stand-alone input type is selected, a menu of available files is displayed. Once a file is selected, a menu of choices for disposition of that file is presented identical to the one in Figure 53.

Otherwise, if a **target** or **threat(s)** input type was selected, a menu like the one in Figure 65 is displayed. If the user selects **entire directory**, he will be prompted for a name for the new directory. Otherwise, if **individual file** is selected, a menu of available files will be presented. Once a file is chosen, the procedure is identical as for selecting a stand-alone file type, and mirrors the procedure for exporting session files (see Section 4.2.1.1, **Analysis Configuration – Sessions**).

4.2.1.3 Configuration and Analysis

Configuration parameters can be reviewed by selecting **show configuration** and can be reset (zeroed out) by selecting **reset configuration** from the menu shown in Figure 47. The configuration of a run is completed by selecting the **analyze** menu entry, also from the "Analysis Configuration Submenu". If the user has chosen *immediate processing*, the computation will begin at this time, otherwise, if *deferred* or *batch processing* was selected, no computation will begin until the session has ended.

4.2.1.4 Access List Configuration

An access list describes which users may access a project and what their level of privileges will be for that project.* This list resides in a file in the project directory and is generated automatically when a new project is created. Initially, only the person who created a project can modify this file. Figure 68 describes, by example, the format of the access list file; like any other input file, this file can only be changed by importing a new copy on top of the old one.

```
# Access list for the Flintstones project
list trustees          fred,wilma
list dependents        pebbles,bambam
owner                  trustees
write                  barney,betty
read                   dependents,dino # watch these guys!
```

Figure 68. Access List File (example)

Comment lines and embedded comments are supported as described in the **Analyst Supplied Input Files** section under **Input File Syntax**. There are 4 possible commands available: **owner, write, read,** and **list**. A command must be the first word on a line; it can only be preceded by spaces and/or tabs. Commands and their associated arguments must not span multiple lines. A command is separated from its first argument by one or more tabs and/or spaces. Each of these commands requires a list of users as an argument; any list entry must be separated from the next by a one or more commas, spaces and/or tabs.

* The MVES administrator is not subject to the restrictions imposed by the access list mechanism.

For convenience in specifying lists of users, the **list** command is available. This command takes a name as the first argument, followed by one or more spaces or tabs and a list of users. For all commands, a list may contain any combination of other lists or users with two caveats: a list must be defined before it is referenced, and a list may reference another list to only one level (a list referenced by another list can not itself reference a list).

The **owner** command lists any users who may modify the access list or perform *file administration* on the project (see Section 4.2.3, **File Administration**). Write access is necessary to run an analysis and implies read access. Read access is sufficient for exporting files or directories, postprocessing results, or displaying files or directories in a project. Ownership implies write permission which, in turn, implies read permission. In the example above, **fred** and **wilma** are owners, **fred, wilma, barney,** and **betty** have write access, and **fred, wilma, barney, betty, pebbles, bambam,** and **dino** have read access.

All commands are optional, their absence denotes *world* access which means unrestricted access to any user. The default access list for a newly created project designates the creator as the sole owner, writer and reader; they must modify the access list to make the project available to other users. The access list can by modified by selecting the **access list configuration** entry of the menu in Figure 47.

Figure 69. Access List Configuration Submenu

The menu in Figure 69 will then be presented. Selecting the **display** entry will send an interpretation (the proper semantics applied) of the access list file to the text buffer. The **import** and **export** entries of the menu give rise to menus similar to the importing and exporting menus for input files explained previously; however, since they pertain only to access list files they are more specific. For instance, the name of the installed file, *.access*, is always the same. If the **import** entry is selected, a menu identical to the one in Figure 63 will be presented. Choosing **from explicit location** will prompt the user for the path name of an access list file to import. Selecting **from other projects**, will present a menu of projects for which the user has read access. Choosing a project is sufficient for the user interface to locate the imported file, however, the user will be asked for confirmation before overwriting an existing access list file. The final entry in the menu in Figure 69 presents a submenu identical to the one in Figure 53 for exporting session files.

4.2.2 Results Interpretation

Since MUVES stores results in MUVES-specific format and within the project file hierarchy, data within these results files can be *interpreted* or *postprocessed* to create summaries or graphical displays. Examining MUVES results files is accomplished by selecting the **results interpretation** entry from the main menu (Figure 43). The user is first presented with a menu of available projects. Once a project is selected the results type must be chosen from a menu as illustrated in Figure 70.

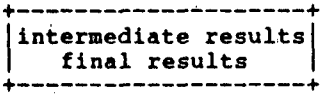
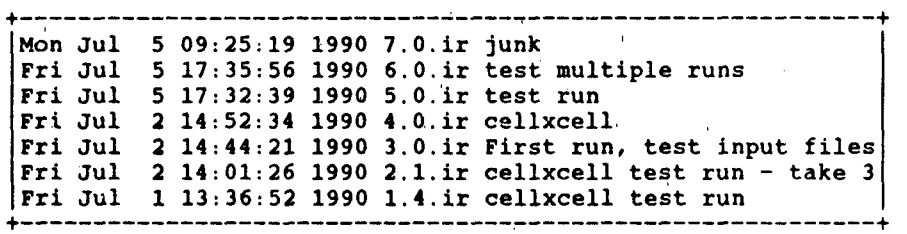


Figure 70. Type of Results Submenu

Picking either **intermediate results** or **final results** will display a menu of the available intermediate or final results files in the current project.



or

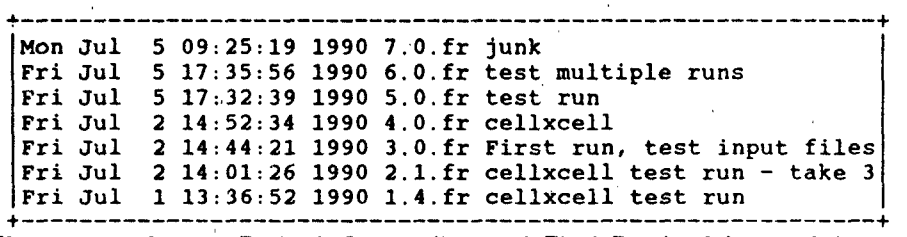


Figure 71. Current Project's Intermediate and Final Results Selection Submenu

Figure 71 illustrates how intermediate/final result files are indicated on the selection menu. Results file menus are similar to session file menus; they include the date and time of last modification (when the run completed), followed by the file name and the title of the run. Note that results files are named from the run number suffixed with the file type ".ir" or ".fr" (intermediate or final results, respectively).

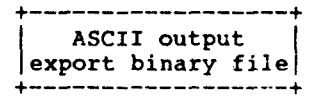


Figure 72. Intermediate Results Postprocessors Submenu

If an intermediate results file is selected from the first menu illustrated in Figure 71, the menu illustrated in Figure 72 will appear. Selecting **ASCII output** executes a postprocessor which formats the chosen MUVES binary-coded intermediate results file into readable form (See Section 5.1.1, **Postprocessors, ASCII conversion**, for details on the output of this postprocessor). The ASCII output can be "exported", as described earlier (see the "Export Submenu", Figure 53). The binary file can also be directly exported with the **export binary file** entry. Once copied out of the MUVES file hierarchy, the user could execute his own postprocessor on the file.

There are currently three postprocessors for final results files. The menu in Figure 73 is used to access the final results postprocessors. The final results file can be displayed in whatever units of measure the user wishes. The entry **presentation units** is used to select the desired units of measure. The following three entries in the menu are for executing postprocessors. The first of these, **color silhouette**, generates a colored cell plot (see Section 5.2.2, **Postprocessors, Color Silhouettes**).

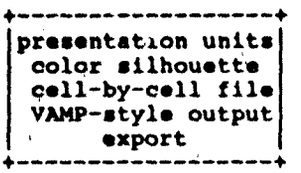


Figure 73. Final Results Postprocessors Submenu

Selecting **cell-by-cell file** generates a table of vulnerability numbers indexed by horizontal and vertical grid coordinates (see Section 5.2.3, **Postprocessors, Cell-by-Cell Results**). The third postprocessor is accessed by the entry **VAMP-style output**. This entry produces VAMP-style probability summary tables, Individual Unit Action (IUA) files and View average files (see Section 5.2.1, **Postprocessors, Compartment-Style Probability Outputs**). The last menu entry, **export**, is for exporting final results files (see the "Export Submenu", Figure 53). The three postprocessor submenus allow the user to set the various options for postprocessor use; each option in a submenu has a help message that explains the option. The user is encouraged to read the **Postprocessor** section, which explains each postprocessor's inputs and outputs in detail.

4.2.3 File Administration

Cleaning up unwanted project files, archiving successful analysis runs, and managing project files is accomplished from the **file administration** entry of the main menu (Figure 43). Upon selecting **file administration**, a menu of projects for which the user has ownership privileges is presented.

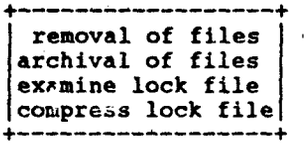


Figure 74. File Administration Submenu

After a project is selected, the menu in Figure 74 is displayed.

4.2.3.1 Removing Project Files – Individual Files

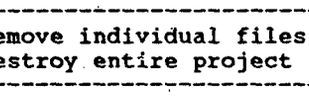


Figure 75. Removal of Files Submenu

To police unwanted files, the user selects the **removal of files** menu entry. The next menu (Figure 75) gives the user two choices: selecting individual files for removal, or destroying an entire project. Choosing the first alternative, the user will be presented with the menu displayed in Figure 76. Selective file removal allows the user to build a list of unwanted files, then remove the entire list as a separate and irrevocable operation. This two part procedure provides some safeguard against removing files unintentionally. To build the list, the user selects **edit list of files** from the menu in Figure 76. The user can then add or subtract files from the file removal list with the resultant menu depicted in

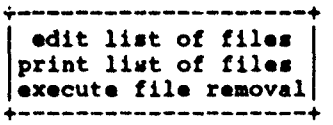


Figure 76. Remove Individual File Submenu

Figure 77. Edit List of Files Submenu

Figure 77. Selecting **add files** produces cascading submenus containing lists of directories and files within those directories that the user may mark for removal. First the general category of files to add to the list must be selected. The menu in Figure 78 shows the choices: input files, results files, and session files.

Figure 78. File Removal Category Submenu

Figure 79. Inputs Submenu

Selecting **inputs** produces the menu in Figure 79. The names in this menu correspond to the input file types described in Table 9. Selecting any of the entries in Figure 79 produces submenus with entries consisting of input files or directories the user has imported into the current project. (See Section 4.2.1.2, **Analysis Configuration – Input Selection**.) If a directory is selected, the user will be presented with a menu of files within that directory. For instance, the menu depicted in Figure 80 is generated from the *sample_ke* threat directory under the *samples* project. The first entry, **-- all files --**, will add all files (**range**, **perf**, and **initial**) to the list of files slated for removal. Otherwise, the user can pick individual entries.

Selecting **results** from the menu in Figure 78 produces a menu of available final results files, intermediate results files, and log files. Choosing **sessions** from the menu in Figure 78 produces a menu of session files. The name of a session file is the session number. Final results files are denoted on the dynamic menu by run number followed by the suffix ".fr". Similarly, intermediate results files are denoted by run number followed by the suffix ".ir". Log messages from all runs in a session are written to one file and appear on the menu denoted by the session number suffixed with ".log". These menus work

TABLE 9. Input File Types

Menu Entry	File Type
ccmap	Component Category Map
dae	Damage Assessment Expression
des	Damage Evaluation Selection
ecurve	Evaluation Modules Inputs
icurve	Interaction Modules Inputs
prop	Component Properties
ray	Reusable Ray
sysdef	System Definition
target	Target Information
threat	Threat Information
view	View

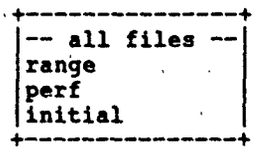


Figure 80. Sample Threat File Removal Menu

the same as those for input files as described in the example in the paragraph above for the *sample_ke* threat. If the file name is not sufficient to enable the user to identify the files, the "file browser", invoked with the **enter-file-browser** command, can be employed to display contents of files in the scrolling-text region.

Once a file has been added to the file removal list, it can be taken off the list by selecting **subtract files** from the menu in Figure 77. This produces a dynamic menu listing the current files slated for removal. As an entry is selected from this menu, it is deleted from the file removal list.

The **print list of files** entry (Figure 76) will send the list of files to the text buffer for the user to examine. To actually dispose of the files in the file removal list, the user should select **execute file removal** (Figure 76). The user will not always be permitted to remove all files in a project. If the file was used in a successful analysis run and is considered necessary for the task of configuration management, it will not be removed until the run has been either *archived* or *unprotected* (see Section 4.2.3.4, **Archiving Project Files**). These files are considered "marked for archival" by **muves**; files that are protected in this manner include session files, final results files, and input files.

4.2.3.2 Removing Project Files – Entire Directories

Removing directories, such as targets or threats, must be accomplished as a two step process. First, all files are deleted by selecting the **-- all files --** entry for the target or threat, and selecting the **execute file removal** entry as described above.

```
Directory "inputs/target/ktank" is empty.
Remove (y or n) [n]:
```

Figure 81. Remove Directory Prompt

Once the directory is emptied in this manner, selecting it once more will trigger a prompt similar to the example depicted in Figure 81. If the user responds affirmatively, the directory will be removed.

4.2.3.3 Removing Entire Projects

Rather than removing a list of files, if the user wishes to remove an entire project, the **destroy entire project** entry of the menu in Figure 75 must be selected. The user will be prompted for confirmation before a project is destroyed in this manner. If any of the files are marked for archival, these will not be removed; however, the project will no longer be usable. If this occurs, the user should use the archival facility described in the next section to archive or unprotect any runs that remain. Once that is done, the **destroy entire project** entry must be selected again, and should run to completion.

4.2.3.4 Archiving Project Files

Archiving analysis runs will produce a permanent record on mass storage which can later be retrieved. To foster the reproducibility and sharing of analysis results, **muves** encourages archiving by marking files for archival that have been used in *successful runs*. Successful runs are any analyses that complete without error. Once marked, files cannot be removed or overwritten by users unless they use the archival facility.

```
+----------+
|   save   |
|unprotect |
+----------+
```

Figure 82. Archival of Files Submenu

The **archival of files** entry of the menu in Figure 74 will display a menu of choices as shown in Figure 82. The **save** entry will display a menu of sessions similar to Figure 50. Selecting a session will bring up a menu of completed runs similar to the one depicted in Figure 51. Selecting a run will initiate the transfer of the associated files to mass storage. If the run was not marked for archival, presumably because it was unsuccessful, the user will be asked for confirmation before **save** can proceed. Once the transfer has completed successfully, a unique identifier for the archive will be reported to the user. It is

incumbent upon the user to record this identifier in his notebook for future reference.†

Alternatively, if the user is certain that the run is not useful, it may be unprotected by choosing the **unprotect** entry from the menu in Figure 82. This results in session and run menus to be presented as above. Unprotecting runs does not remove the associated files; the file removal procedure must be used to accomplish this.

It is vital, from a configuration management prospective, that any analysis runs whose results are published or distributed be archived.

4.2.3.5 Administrating Project Lock Files

For the purpose of tracking files that are currently in use by an analysis or have been marked for archival, **muves** maintains a lock file for each project. This file contains an *in-use count* and an *archive count* of files that are being or have been used in an analysis run. The in-use count represents how many runs are currently using a file. The archive count keeps track of the number of runs that have completed successfully using a file. Whenever a run is archived or unprotected, the archive count for all associated files is decremented. The user may examine this file with the **examine lock file** entry of the menu depicted in Figure 74. It is possible that due to hardware failures or other unforeseen events that this file may have been corrupted or will not reflect reality. For instance, if analyses are running when a computer goes down, in-use counts will not be decremented to show that the analyses are no longer running. If, upon examination of the lock file, the user deems that the file has been corrupted, he should request the assistance of a MUVES administrator to remedy the situation.

Lock files will accumulate entries with zero in-use and archive counts. Since such entries are not informative and cause clutter when displaying lock files, the user may select the **compress lock file** entry of the menu in Figure 74 to remove them.

† Databasing facilities may exist at your site to search for archived runs and retrieve the MUVES identifier, but this software is not currently distributed with MUVES. Ask your MUVES administrator if such a facility is available.

5. Postprocessors

For each analysis run, MUVES produces a final results file which contains the fractional remaining functionalities for each shot in every view analyzed during the run. Analysis selections, such as threat and shot pattern selections, determine the number of remaining functionality sets in these files. Within MUVES, analysts may also specify an option to save intermediate results in a file. Intermediate results files contain various internal values (*e.g.*, geometric normals, residual penetration, *etc.*) used to perform computations during an analysis. Because results files contain a substantial amount of information, interpretation of the data in these files is easier when it is summarized or displayed graphically, *i.e.*, postprocessed.

Postprocessors are part of the MUVES distribution. All postprocessors are stand alone programs and are accessible just like other UNIX tools. Executables will reside in the $MUVES/bin directory. Every postprocessor distributed with MUVES has an on-line manual page. Manual pages for the postprocessors being distributed with MUVES are provided in alphabetical order in an appendix to this report. For convenience, these postprocessors can also be invoked from the MUVES user interface. Menus in the user interface will prompt for information to be passed to the postprocessor.

This section explains, in general terms, how to use these postprocessors and their options, without giving detailed instructions concerning their use. Consult the appropriate postprocessor manual pages for further elaboration. This section also describes the output formats produced by each postprocessor and provides sample outputs.

At this time, the postprocessors distributed with MUVES are designed for compartment model analyses and may not work with other approximation methods. As time permits, additional postprocessors, such as butterfly charts and cell difference plots, will be added. As new approximation methods are incorporated into MUVES, postprocessors for these methods will also be added.

A general comment about units is appropriate at this point. The default behavior for all MUVES postprocessors is to print relevant output values in millimeters. However, since not all analysts wish to work in these units, the postprocessors provide a units option for converting their output to the desired unit system. The arguments to this option must be chosen from among the names or symbols shown in the table below in order to be recognized by any program in the MUVES environment.

TABLE 10. Valid Unit Names

Name	Symbol
feet	ft
inches	in
meters	m
centimeters	cm
millimeters	mm
microns	

5.1 Postprocessors for Intermediate Results Files

The intermediate results files contain a tremendous amount of information recorded during the course of an analysis. To avoid poor computer performance, this information is recorded in binary form; therefore, it is necessary to provide a translator to make the files human readable. At the moment, an ASCII conversion program, called **irdump**, is available to translate intermediate results files to a generic, approximation-method-independent form. In addition, a "compart" method specific program, called **ircompart** is also available. See the **ircompart** manual page for more information.

5.1.1 ASCII conversion

The program for translating binary intermediate results into human-readable form is called `irdump`. Most of the options for this program relate to showing the grid locations of the shots; that is, printing the center coordinates for the grid cell in which a particular shot lies. There are options for specifying the size of the grid cells in cases where a grid was not used to generate the shots (*e.g.*, grouped shots). There is also an option for changing the unit type used for printing the output; ordinarily, length values are printed in millimeters, but the analyst may change this to one of several recognized unit types. The name of the intermediate results file must follow the list of options. `irdump` prints a translation of the specified file to standard output.

The resulting file contains the information computed during the course of the analysis. Most of the information is labeled, although certain portions may be difficult to follow. Specifically, the threat and damage packets are labeled, but their parameters are not; these values are converted to the proper type (real, integer, Boolean, or character string) for easier comprehension. Consult the appropriate approximation method documentation for details of the meaning of each value. In addition, the evaluation portion of each shot contains only those components which have been damaged, that is,

components which are *not* set to the default utility value, as defined by the approximation method.* The figure below displays a portion of a converted intermediate results file created during a compartment model analysis.

```
az 90 al 0    dir < -6.12323e-17 -1 0 >  shot -1671.09 -1570.95  cell -1800 -1800
KE "track edge lt"
        entry < 1671.09 1424.94 -1570.95 > normal < 0 1 0 >
        SAND_OBL        0
        KE_S    HVAP 120 24.3 1 1905 1905 259.6 259.6
        los thickness   2.54004
        wtd normal thickness    0.254004
        normal thickness        2.54004
        obliquity       0
            .
            .
            .
KE "track edge rt"
        entry < 1671.09 -1422.4 -1570.95 > normal < 0 1 0 >
        SAND_OBL        0
        KE_S    HVAP 120 24.3 1 1905 1579.98 259.6 178.574
        los thickness   2.54004
        wtd normal thickness    0.254004
        normal thickness        2.54004
        obliquity       0
KE "MUVES exit paint"
        entry < 1671.09 -1424.94 -1570.95 > normal < 0 1 0 >
        SAND_OBL        0
        KE_S    HVAP 120 24.3 1 1905 1578.86 259.6 178.32
- damage assessment -
        typical/firepower
        typical/mobility
                "track left"    0.99312
                "track right"   0.99312

az 90 al 0      dir < -6.12323e-17 -1 0 >  shot -858.078 -1515.6  cell -900 -180
0
KE "other wheels"
        entry < 858.078 1346.2 -1515.6 > normal < 0 1 0 >
        SAND_OBL        0
        KE_S    HVAP 120 24.3 1 1905 1905 259.6 259.6
        los thickness   228.6
        wtd normal thickness    228.6
        normal thickness        228.6
        obliquity       0
            .
            .
            .
```

Figure 83. Sample from an Intermediate Results File

---

* The default utility value is normally 1.0, but this may be altered within each approximation method. Consult the approximation method appendix for details of the exact meaning of the utility values for the components.

5.2 Postprocessors for Final Results Files

The final results files contain a header describing the analyst's selections for the analysis, followed by a (potentially large) set of information lines, one line for each shot that hit the target. These lines are grouped into views for referencing purposes. Each line contains the three-dimensional coordinates of the shot and the remaining functionalities of the target in each context for that shot.

These files are intended to hold the basic results of an analysis in a form convenient for computer usage and storage. Therefore, although stored as text, they are not especially readable to the average person. Final results files are best examined using the facilities described in this section.

5.2.1 Compartment-Style Probability Outputs

A program is available to convert output from the MUVES "compart" approximation method to the formats produced by older versions of the compartment model. This program is called **siv**, which stands for Summary tables, Individual Unit Action (IUA) files and View averages. This name was chosen because it reflects the three types of output available from **siv**.

siv's primary function is to compute weighted averages of the losses of function for all specified views of the target in a given final results file. The losses of function represent the target's loss in capability after being damaged by a threat. Each view in the final results files must consist of a single "grid" pattern; any other shot pattern will be ignored by **siv**. (Currently, the algorithm cannot process anything other than grids.) In a grid pattern, the vehicle is divided into rectangular areas of equal size called "cells". Each shot is selected from a unique cell and is used to represent the damage caused by firing a shot in that cell. **siv** computes a weight representing the likelihood, for a given aim dispersion, of a random shot striking that cell and combines it with the loss of function for the shot. These weighted losses of functions are summed to form the "most probable" loss of function for the target when attacked from one direction with a particular munition. This process is repeated for each aim dispersion requested by the analyst. These direction-dependent average losses are usually combined using combat weighting functions into a single value indicating the most likely loss of function for the given vehicle when attacked in a battlefield environment 15 with that munition.

The weight for each cell in a view is calculated using a Gaussian approximation integrated over the boundaries of the cell. This represents the aim dispersion for a munition directed against the target. The standard deviations for the aim dispersions are specified as concentric rings around the desired aim point for each view. The size of these standard deviations is specified using two values. The first is the number of dispersions desired per view, while the second is the interval between the standard deviations. Thus, specifying 10 for the first value and 100 for the second will produce standard deviations of 100mm, 200mm, 300mm, and so on to 1000mm. In this application, the vertical and horizontal standard deviations are equal.

The unit type used for the standard deviation depends on a selection made by the analyst; they will be in millimeters if the analyst makes no selection. The units selection also applies to the offsets in the "view

information file" described below.

Some information required by **siv** is view specific and cannot be input using command line options. Such values are entered using the view information file (not to be confused with the view file described in Section 3.3.5, **Analyst Supplied Input Files, View File**). For each viewing direction in the final results file, one may specify a corresponding definition in the view information file. This definition includes the view direction (azimuth and elevation), a weight relative to other views in the file, plus horizontal and vertical offsets for the center of the aim dispersions. For instance, to set a weight of .15 for a side view (90 azimuth, 0 elevation) of the target with a horizontal offset of 100mm and a vertical offset of -200mm, one would enter the following line:

```
90   0    .15   100   -200
```

These weights and offsets only apply to the specified view; each view has its own values for these parameters.

In addition, the analyst may also wish to specify one or more defilade conditions for analysis. To define a defilade condition for a given view, one should enter a line containing the word "defilade" before the next view is defined. This word must be followed by a vertical cutoff value and another pair of horizontal and vertical offsets. For example:

```
defilade   100   0   400
```

The cutoff value is used to eliminate shots from consideration in the averaging; in this case, any shot whose vertical coordinate is less than 100mm will be ignored in the averaging. The defilade offsets are used *only* for the defilade condition; they are not combined with the offsets specified for the view. Although the defilade conditions are averaged separately from the fully-exposed views, the defilade condition inherits its averaging weight from the view definition.

It is permissible to enter more than one defilade condition in the view information file. In this case, one would receive average weights for the fully exposed case and for each of the specified defilade conditions. However, one should be certain that each view has the same associated defilade conditions as the other views. The combat-weighted averaging will be inconsistent if this is not the case, since not all views will be represented for all defilades. A portion of a real view information file is shown in the figure below. This particular file uses inches for the cutoff and offsets (which would require setting the units option when running **siv**).

```
#    defilade and adjust values are in inches
#
#azim          elev          weight          xadj    yadj
#defilade      -             cutoff          xadj    yadj

0.0            0.0           0.273           0.00    -12.00
defilade                     4.0             0.00    18.00

30.0           0.0           0.201           12.00   -12.00
defilade                     4.0             12.00   18.00
```

Figure 84. Sample View Information File Entry

The weights for all views in a view average file should sum to 1.0, since they represent the fraction of shots that are likely to come from each direction. However, this is not enforced within **siv**; the weights will be used exactly as they are entered in the file.

siv has several other options available for determining further averaging parameters and specifying the output files.

siv performs its calculations for a single threat. If there is more than one threat in the final results file, the name of the desired threat should be specified in the option list. If none is specified, the first threat listed in the results file header will be used.

Many weapons have an inherent bias associated with them. One may specify both a horizontal and vertical bias to account for this. The bias will be applied to all views of the target and will be added to the offset specified for each view in the view information file. The values should be in the same units as the offsets.

By default, the target center is chosen to be the target origin. The weapon bias and aim point are offset from the target center to determine the center of the aim dispersions. The centroid of the view (*i.e.* the center of presented area) may also be specified as the target center. The weapon bias and aim point will then be applied to the coordinates of the centroid in the same manner.

Once the view averages are computed, the information is written in the format(s) requested by the user. If only one or two types of output are required, they may be specified in the option list. Otherwise, **siv** will produce all three types of output. The output files will all have the same user-specified prefix. Each file will have a unique suffix added on to that prefix. The suffixes are ".sum" for summary files, ".iua" for IUA files, and ".avg" for view average files. If no output prefix is specified, all of the requested files will be written to standard output.

In the output files which follow, the values are often referred to as "probabilities of kill". For all kill types except the catastrophic kill, this label is a misnomer but is retained for consistency with historical output formats. In actuality, these values are averages of "losses of function" for each shot in a view and should not be treated as true probabilities.

The summary tables are intended to provide a quick check on the overall vulnerability of a vehicle. They show the combat-weighted average losses of function of the target for all views under various conditions of aim dispersion and defilade. The threat and target are indicated using the names from the final results file. The dispersion columns show the size of the aim dispersions; in the figure below, the

values are in inches. The kill types are labeled across the top with standard abbreviations. MNP stands for "mobility kill with no perforation"; in this example, that mission function was not provided in the final results file. The other abbreviations stand for "mobility"(M), "firepower"(F), "maximum mobility/firepower"(MF), and "catastrophic"(K) kills.

A sample of a summary table for probabilities of kill given a hit is shown in the figure below. There are separate summary tables for the fully-exposed condition and each defilade condition. There are also corresponding tables for probability of kill given a shot (not shown); these look identical except that the word "shot" is substituted for the word "hit" (they usually contain different values, as well).

```
Probability of kill given a hit for various values of dispersion
          (averaged over attack angles)
Distribution of attack angles weighted as specified in input

          Projectile  pl165  vs. Vehicle  tank1

                    Fully-Exposed Target

  Aim Point          Dispersion              Probabilities of Kill
horiz     vert     horiz     vert      MNP      M        F       MF       K

 0.00     0.00    12.00     12.00    0.000   0.668    0.671   0.698   0.201

 0.00     0.00    24.00     24.00    0.000   0.590    0.580   0.625   0.158

   .
   .
   .

 0.00     0.00   120.00    120.00    0.000   0.420    0.327   0.462   0.089

              -- infinity --        0.000   0.397    0.301   0.445   0.078
```

Figure 85. Summary Table Sample

A portion of an IUA file is shown in the next figure. This is a compact format designed primarily for input to a variety of computer programs. The columns are labeled across the top of the file and have the following meanings. VC and PC stand for vehicle code and projectile code respectively. These may be labeled with four digit integers using input options. If no values are specified for these codes, the columns will be left blank, as in the example. R stands for range and indicates the range in meters from the threat to the target. Range is only applicable to kinetic energy munitions, so for shaped charge munitions, the value 9999 is printed in this column; this is done for historical reasons. E is for exposure; the defilade conditions are labeled 1 through N, depending on how many there are in the view information file. The fully-exposed case comes last in the file and has the highest value in this column. D indicates aim dispersions, which are numbered 1 through N+1 where N is the number of dispersions specified in the input options. The last value in this column is for the "infinite" dispersion, which is the equally weighted average over all cells. K refers to the kill type. These are mobility (indicated by a 1), firepower (2), maximum mobility/firepower (3), and catastrophic(4), respectively. The numbers across the top are the azimuths for the attack aspects listed in the final results file, and the last column is the overall average, weighted according to the view information file.

VC	PC	R	E	D	K	0	30	60	90	120	150	180	AVG
		0	1	1	1	0.182	0.685	0.751	0.783	0.862	0.755	0.941	0.597
		0	1	1	2	0.359	0.747	0.792	0.857	0.917	0.768	0.921	0.686
			1	1	3	0.359	0.747	0.792	0.857	0.917	0.801	0.942	0.687
		0	1	1	4	0.116	0.024	0.289	0.000	0.268	0.607	0.898	0.146
		0	1	2	1	0.124	0.530	0.601	0.658	0.769	0.658	0.751	0.482
		0	1	2	2	0.202	0.612	0.628	0.717	0.809	0.648	0.709	0.539
		0	1	2	3	0.202	0.613	0.631	0.717	0.815	0.692	0.755	0.542
		0	1	2	4	0.064	0.023	0.319	0.045	0.309	0.434	0.659	0.146
		0	1	3	1	0.098	0.437	0.545	0.602	0.703	0.615	0.628	0.426
		0	1	3	2	0.154	0.528	0.574	0.659	0.731	0.598	0.583	0.478
		0	1	3	3	0.154	0.535	0.584	0.661	0.746	0.646	0.633	0.485

Figure 86. Beginning of an IUA File

A portion of a view average file is shown in the next figure. The view average file contains the weighted averages for each of the views in the final results file. It shows both the probability of kill given a shot and probability of kill given a hit for each of the aim dispersions specified by the analyst. It also shows probabilities of hit for various portions of the target, including the whole target (TGT), armor (ARM), internal volume (IV), and internal volume perforated (IVP).† If the appropriate values are not present in the final results file, the corresponding column will be set to 0.0. This can be seen in the ARM, IVP and MNP columns in the following example. The file contains similar tables for each combination of defilade condition and attack aspect. Note that Figure 87 shows two rows of data for each aim point and dispersion. The first data row contains probabilities of hit and kill given a shot. The second row contains probabilities of hit and kill given a hit on the target.

† In order to include these items in the final results file, one must create and select appropriate mission functions. MUVES prototypers have used the "threatened" environment to represent these cases, with mission functions such as "internal_vol", "crew_air", *etc.*. Apparently, the armor column is an anachronism and will be replaced with more relevant information in future versions of this code.

```
Probability of kill given a shot and probability of kill given a hit on the
target for various values of dispersion and various angles of attack
(Assuming a normal distribution of hits around the center of impact)

          Projectile  p1165  vs. Vehicle  tank1.ke
          Aspect Angle of 0.0 degrees Azimuth 0.0 degrees Elevation

                                  Fully-Exposed Target
          H Bias = 0              V Bias = 0

  Aim Point        Dispersion      Probabilities of Hit    Probabilities of Kill
 horiz     vert    horiz    vert   TGT   ARM   IV   IVP   MNP    M     F    MF     K

 0.00    -12.00   12.00   12.00  0.994 0.000 0.677 0.000 0.000 0.315 0.347 0.349 0.042
                                 1.000 0.000 0.682 0.000 0.000 0.317 0.349 0.352 0.042

 0.00    -12.00   24.00   24.00  0.884 0.000 0.516 0.000 0.000 0.221 0.234 0.256 0.030
                                 1.000 0.000 0.584 0.000 0.000 0.250 0.265 0.290 0.034

 0.00    -12.00   36.00   36.00  0.709 0.000 0.340 0.000 0.000 0.151 0.147 0.173 0.018
                                 1.000 0.000 0.479 0.000 0.000 0.212 0.207 0.245 0.025

 0.00    -12.00  120.00  120.00  0.125 0.000 0.044 0.000 0.000 0.021 0.018 0.024 0.002
                                 1.000 0.000 0.350 0.000 0.000 0.169 0.146 0.191 0.017
```

Figure 87. View Average File

5.2.2 Color Silhouettes

There is a program available for creating colored cell plots of target vulnerability estimates; it is called **colorsil**, which is an abbreviation for "color silhouette". This program provides a quick way to obtain general information about the vulnerability of a target. It is also a good presentation tool when discussing large quantities of results for a given target.

colorsil reads one view from a final results file and converts each shot in the view to a colored square on a graphics display. Each square is color-coded to one target functionality for that shot. (Note that there will usually be more than one functionality computed for the target at each shot location.) These squares will generally form a silhouette of the target wherein the vulnerable parts of the target are clearly visible. The color information is written in BRL-CAD [10] pix format to a device specified by the user. The images may then be manipulated using BRL-CAD tools, if desired.

The analyst must select the desired view from among those in the results file header; if not, the first view listed will be used. If there is more than one threat in the final results file, one may specify which is desired by naming the threat and giving the associated range (if appropriate). If a threat is not specified, the first threat listed in the final results header will be displayed. Another selection that must be made from the final results file is the appropriate environment/mission pair (context) to be displayed. The default behavior is to use the first environment/mission pair listed.

Each shot in the view has a value associated with it representing the target's fractional remaining functionality in the desired context as a result of the damage done by the threat. `colorsil` converts this value to a color based on the following table:

TABLE 11. Remaining Functionality Color Conversion Key

Value	Color
0.0	red
.1	pink
.2	orange
.3	yellow
.4	green
.5	dark green
.6	turquoise
.7	light blue
.8	blue
.9	blue-grey
1.0	white

Ordinarily, any value within plus or minus .05 of a value in the table is converted directly to the corresponding color. One may specify that colors be interpolated (or blended). This causes the values to be converted by blending the colors for the two boundary values. Given the limitations of human color perception, this option is not generally useful, but is retained for historical reasons.

Alternatively, one may wish to view the image on a black and white display. There is a monochrome option which scales the grey level according to the magnitude of the value. 1.0 still converts to white, while 0.0 converts to black. Others values are distributed evenly between these extremes. To give an idea of the type of output produced by `colorsil`, a black and white image is shown in the next figure.

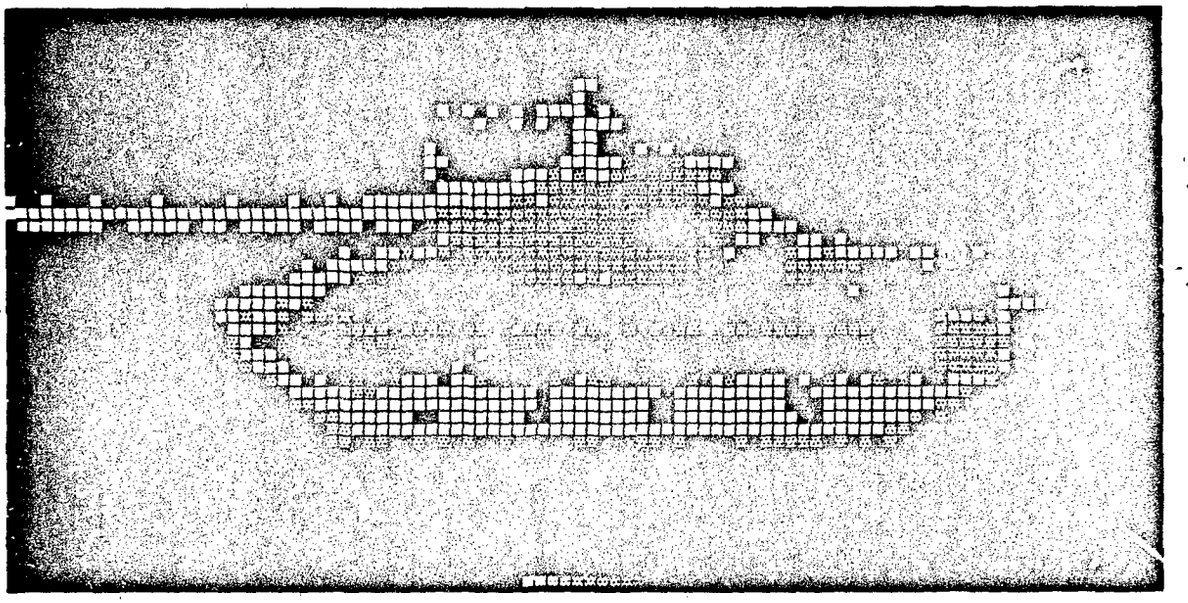


Figure 88. `colorsil` Black and White Image

`colorsil` has a variety of other options for configuring the displayed image.

Any shot pattern or group of shots may be displayed using `colorsil`. The firing point for each shot will be converted to two dimensional coordinates and the result will be used as the center of a colored square in the display. This means that some shots may be partially or wholly obscured by other shots in the view. If the analyst wishes to shift the squares so that they appear in regular rows and columns, he may set an option to do so; this is called "binning" the shot locations. If the shots were produced using a "grid" pattern, they will shifted according to the parameters of that grid; otherwise, the analyst may set the parameters of the desired grid pattern in the option list. This is especially useful when viewing a shot group that was originally produced from a grid pattern, as is often done for comparisons with results from outside the MUVES environment. Other patterns may leave some shots completely obscured by shots which end up in the same bin.

By default, `colorsil` puts a one pixel wide space between the cells; however, the size of this space may be altered or set to zero (which removes spaces altogether). Another default is to show a color key containing all of the major colors at the bottom of the display. This behavior may also be disabled using a

configuration option. There is also an option to superimpose grid plane axes over the cell plot. This is very helpful for finding the grid plane coordinates of a particular shot.

Options exist to change the dimensions (*i.e.*, total size) of the image and the size of the cells in the display. No attempt is made to contain the cells within the specified screen resolution; instead, they are cropped if they do not fit into the display. One may also offset the image to a different location in the display space. The same cropping limitation applies to this option.

Finally, there is the choice of frame buffer device. This may be done indirectly by setting the environment variable FB_FILE**, or directly by using an input option. This will usually be a color display monitor, although it may also be a file. Caution is advised when writing to a file, since such files tend to be rather large; creating several such files will quickly occupy a significant percentage of available disk space.

5.2.3 Cell-by-Cell Results

Cellxcell generates Compartment-style cell-by-cell outputs from a MUVES final results file. Cell-by-cell planar coordinates and loss of functionality/probability values are produced for all environment/mission pairs in the final results file. In the special case where the environment/mission queue in the final results file contains both "firepower, typical" and "mobility, typical" contexts, **cellxcell** automatically outputs an environment/mission column for each shot location that contains the maximum of the "mobility, typical" and "firepower, typical" values. This column will always be in the column after the latter of the two mission/environment pairs.

In a MUVES final results file, a set of remaining functionalities/probabilities is described by the analysis approximation method, the target-threat combination, and the shot specification. Each set output by **cellxcell** is labeled with target, threat, target-threat range (if applicable), azimuth, and elevation information. A sub-heading is used to identify the X,Y coordinates and contexts.

If **cellxcell** is invoked without a target, threat, or view selection, **cellxcell** outputs all "compart" cell-by-cell results for single shots, a group of single shots where all shots have a common azimuth and elevation, and patterns of shots (*e.g.*, grid, bivariate gaussian). By historical definition, cell-by-cell planar coordinates are the shot coordinates expressed as center of cell coordinates. However, for non-grid patterns, *e.g.*, a bivariate gaussian shot pattern, center of cell coordinates are not meaningful. In this later case, **cellxcell** always outputs the actual shot coordinates (*i.e.*, planar coordinates). **Cellxcell** has an option to output planar shot coordinates in lieu of cell-by-cell coordinates for groups of shots and grid shot patterns.

** This variable may be set in the shell to the analyst's choice of display device. The standard form of the value is *machine_name:device_name*. Thus, a Bourne shell user writing to an SGI workstation named "whatever" would use the following: `FB_FILE=whatever.brl.mil:/dev/sgil export FB_FILE`.

If **cellxcell** is invoked with target, threat, or view selections, output is produced based on these selections. **Cellxcell** may also be invoked with the desired output units for X and Y coordinates. If no units are specified, the default is millimeters.

```
X,Y Units: millimeters

Key   Mission/Environment
=======================
(1): firepower, typical
(2): catastrophic, typical
(3): mobility, typical
(4): maximum(mobility, firepower), typical
(5): internal_vol, threatened
(6): gunners primary sight, threatened


TARGET: tank THREAT: p9999 RANGE: 1000.00 AZIMUTH: 0.00 ELEVATION: 0.00
    ~X       ~Y       (1)      (2)      (3)      (4)      (5)      (6)
  200.00   200.00    0.939    0.876    0.700    0.939    0.600    0.650
    0.00     0.00    0.899    0.776    0.700    0.899    0.600    0.450

TARGET: tank THREAT: p1111 RANGE: 1000.00 AZIMUTH: 0.00 ELEVATION: 0.00
    ~X       ~Y       (1)      (2)      (3)      (4)      (5)      (6)
  200.00   200.00    0.928    0.830    0.600    0.928    0.555    0.550
    0.00     0.00    0.775    0.776    0.600    0.775    0.555    0.450
```

Figure 89. Sample cellxcell output

Figure 89 contains a sample cellxcell output for an analysis with a fictitious tank and two threats.

References

[1] Paul H. Deitz and Aivars Ozolins, "Computer Simulations of the Abrams Live-Fire Field Testing," Ballistic Research Laboratory, Memorandum Report No. 3755, May 1989.

[2] C. L. Nail, T. E. Bearden and E. Jackson, "Vulnerability Analysis Methodology Program (VAMP): A Combined Compartment-Kill Vulnerability Model," Computer Sciences Corporation Technical Manual, CSC TR-79-5585, developed under BRL Contract DAAK40-78-D-0005 (Order No. 0029), October 1979.

[3] C. L. Nail, "Vulnerability Analysis for Surface Targets (VAST) - An Internal Point-Burst Vulnerability Assessment Model - Revision I," Computer Sciences Corporation Technical Manual, CSC TR-82-5740, August 1982.

[4] Douglas A. Ringers and F. Tyler Brown, "SLAVE (Simple Lethality and Vulnerability Estimator) Analyst's Guide," US Army Ballistic Research Laboratory Technical Report ARBRL-TR-02333, June 1981.

[5] Aivars Ozolins, "Stochastic High-Resolution Vulnerability Simulation for Live-Fire Programs," *The Proceedings of the Tenth Annual Symposium on Survivability and Vulnerability of the American Defense Preparedness Association*, held at the Naval Ocean Systems Center, San Diego, CA, May 10-12, 1988.

[6] Phillip J. Hanes, Karen Ross Murray, Douglas A. Gwyn and Helen R. Polak, "An Overview and Status Report of MUVES (Modular UNIX-based Vulnerability Estimation Suite)," Ballistic Research Laboratory Report, Memorandum Report No. 3679, July 1988.

[7] Tom De Marco, *Structured Analysis and System Specification*, New York, NY: Yourdon, Inc., 1978.

[8] Edward Yourdon, *Managing the System Life Cycle*, New York, NY: Yourdon Press, 1982.

[9] Keith A. Applin, Michael J. Muuss and Robert J. Reschly, "Users Manual for BRL-CAD Graphics Editor MGED," US Army Ballistic Research Laboratory, Draft copy, 6 October 1988.

[10] Michael Muuss, Phillip Dykstra, Keith Applin, Gary Moss, Paul Stay and Charles Kennedy, "The Ballistic Research Laboratory CAD Package Release 3.0 - A Solid Modelling System and Ray-Tracing Benchmark," edited by Donald Merritt, SECAD/VLD Computing Consortium, US Army Ballistic Research Laboratory, 1 October 1988.

[11] David L. Rigotti, Paul H. Deitz, Donald F. Haskell, Michael W. Starks, Daniel P. Kirk, Orlando T. Johnson, John R. Jacobson, William Kokinakis, J. Terence Klopcic and Gilbert A. Bowers, "Vulnerability/Lethality Assessment Capabilities - Status, Needs, Remedies," Ballistic Research Laboratory, Special Publication 74, December 1988.

[12] Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman, *Compilers - Principles, Techniques, and Tools*, Reading, MA: Addison-Wesley Publishing Company, 1988.

[13] Robert DiPersio and Julius Simon, "Theory of Residual Penetration by Ideal Shaped Charge Jets," US Army Ballistic Research Laboratory, Report No. 1313, March 1966.

[14] Robert DiPersio, Julius Simon and Alfred B. Merendino, "Penetration of Shaped-Charge Jets into Metallic Targets," US Army Ballistic Research Laboratory, Report No. 1296, September 1965.

[15] Wilbert J. Brooks and W. Donald Johnson, "Horizontal Attack Angle Distributions for Armored Vehicles," US Army Materiel Systems Analysis Activity, Technical Report No. 481, November 1989.

[16] L. W. Bain Jr. and M. M. Reisinger, "The GIFT Code User Manual; Volume I, Introduction and Input Requirements," Technical Report, BRL Report No. 1802, July 1975.

[17] Robert L. Kirby, "Appendix F, A New Crew Compartment Algorithm for the Compartment Vulnerability Model," US Army Ballistic Research Laboratory, version of 15 May 1987, unpublished.

[18] Aivars Ozolins, "Compilation of VAMP Damage Evaluation Procedures, Target-Threat Interaction Algorithms, and Associated Numerical Approximations," US Army Ballistic Research Laboratory, 5 March 1987.

[19] Robert DiPersio and Julius Simon, "The Penetration-Standoff Relation for Idealized Shaped-Charge Jets," Ballistic Research Laboratories Memorandum Report No. 1542, February 1964.

[20] Steven B. Segletes, "A Model of the Effects of Transverse Velocity on the Penetration of a Shaped-Charge Jet," US Army Ballistic Research Laboratory, Memorandum Report BRL-MR-3409, November 1984.

Appendix A: Online Manual Pages

INTENTIONALLY LEFT BLANK.

NAME

colorsil – produce color plots from MUVES final results

SYNOPSIS

colorsil [options...] final_results_file

DESCRIPTION

colorsil reads a view from a MUVES final results file and produces a color plot representing the values for a single environment/mission pair within that view. The information is written out to the specified display in BRLCAD pix format. The image may then be manipulated using BRLCAD tools.

colorsil produces a plot of colored squares (or "cells") where each cell represents one shot in the results file. The cells are centered on the actual shot location within the display space, so that each cell shows the exact location of the damage producing hit. For any shot pattern produced using a randomizing technique, the end result can be somewhat confusing, so an option is provided to "bin" the cells into uniform rows and columns. This procedure may require supplementary information in the form of bin sizes, depending on the pattern used to produce the view being displayed.

In order to select the view, the analyst must give *colorsil* a number corresponding to the position of the desired view in the results file header. If there is more than one threat in the final results file, one may specify which is desired by naming the threat and giving the associated range (if appropriate). If these are not specified, the first threat listed in the final results header will be displayed. Another selection that should be made from the final results file is the appropriate environment/mission pair to be used. If this is not specified, *colorsil* will display the first pair listed in the results file.

Each shot in the view has an associated value representing the target's remaining functionality for the desired environment and mission function as a result of the damage produced by interaction with the threat. *colorsil* converts this value to a color based on the following table:

Value	Color
0.0	red
.1	pink
.2	orange
.3	yellow
.4	green
.5	dark green
.6	turquoise
.7	light blue
.8	blue
.9	blue-grey
1.0	white

Ordinarily, any value within plus or minus .05 of a value in the table is converted directly to the corresponding color. One may specify that colors be interpolated (or blended). This causes the values to be converted by blending the colors for the two boundary values. Given the limitations of human color perception, this option is only useful in certain instances.

One may also specify that the cells be displayed in black and white; in which case 0.0 will appear black and 1.0 will appear very light grey, with other values evenly shaded between these extremes. This is useful for display on various monochrome terminals and printers.

An important consideration for display purposes is the size of a pixel in terms of the target. This determines the density of the colored squares on the screen. This parameter is not set directly; it is set implicitly using a couple of different options, depending upon the type of sampling used for

the view to be displayed. For a "grid" pattern, this is set using the combination of cell size and spacing in each direction. Larger values for spacing will produce a lower cell density in the image. The spacing will be quietly ignored if the pattern is not a "grid". For other patterns, the exploded view factor is used to determine the cell density. A lower factor will result in a lower number of cells displayed in the image, while decreasing the cell size will cause the cells to look smaller. This factor can also be used with grid patterns, but is not as precise as setting the spacing explicitly.

Options

-b	Use black-and-white scale for cell display.
-c	Interpolate colors between standard values.
-d	Dither display locations if grid points were randomized. "Grid" displays will usually show squares aligned with the cell boundaries. However, the user may specify that they be centered on the true hit location.
-e *environ*	Select environment to be displayed from the final results.
-f *mission*	Select mission function to be displayed from the final results.
-g	Overlay grid-plane axes on cell image.
-h *height*	Set the pixel height of cells displayed in the image.
-i *hres*	Set horizontal screen resolution of the image.
-j *vres*	Set vertical screen resolution of the image.
-k	Do not show color key at the bottom of the display.
-m *method*	Select the approximation method from among those in the final results file.
-o	Overlay colored cells on previously existing image.
-p *threat*	Select the threat from among those available in the final results file.
-q	Turn on testing output. This will produce several lines of diagnostic output for each cell, so it is not recommended for long files.
-r *range*	Select the range from among those in the final results. It is possible that there will be no ranges for the threats represented in a given final results file, in which case this option has no effect.
-s *hspace*	Set the horizontal spacing between displayed cells in a "grid" display. By default, this is set to 1 pixel. Setting it to 0 will leave no spaces, while larger values will produce more space between displayed cells.
-t *target*	Select the target from among those available in the final results file.
-u *vspace*	Set the vertical spacing between displayed cells in a "grid" display. By default, this is set to 1 pixel. Setting it to 0 will leave no spaces, while larger values will produce more space between displayed cells.
-v *view_num*	Select the viewing aspect from the final results file. This is done using the number of the desired view in the sequence specified for the analysis (starting at 0).
-w *width*	Set the pixel width of cells displayed in the image.
-x *hoffset*	Move all cells the given number of pixels in the horizontal direction.
-y *voffset*	Move all cells the given number of pixels in the vertical direction.

-s *factor*	Set the exploded view factor. An exploded view factor greater than one separates the cells without changing their size. If it's less than one, it will crowd them together; if they are too close together, they will begin overlapping.
-F *framebuffer*	Display device or file name for graphics output.
-G	Move shot locations to regular rows and columns (*i.e.* cells) for display. This process is called "gridding" or "binning" the shot locations.
-H *cell_hgt*	Set the cell height for gridding the shot locations.
-O	Center the display on the origin of the target coordinate system. The usual behavior is to use the center of the bounding rectangle.
-U *units*	Set the units for interpreting the -H and -W flags.
-W *cell_wid*	Set the cell height for gridding the shot locations.

SEE ALSO

muves(1)

MUVES Analysts' Guide

The BRLCAD Package — A Solid Modeling System and Ray Tracing Benchmark

AUTHOR

Jeff Hanes, BRL/VLD/VMB

BUGS

A file that is not a MUVES final results file will usually cause *colorsil* to dump core.

NAME

cellxcell – compart cell-by-cell results from MUVES final results

SYNOPSIS

cellxcell [-a] [-p threat [-r range]] [-t target] [-u units] [-v view_no] [-H cell_hgt] [-W cell_wid] final_results_file [outfile]

DESCRIPTION

Cellzcell generates Compartment-style cell-by-cell outputs from a MUVES final results file. Cell-by-cell planar coordinates and loss of functionality (LOF) values are produced for all environment/mission pairs in the final results file. Each view in the output is labeled with target type, threat type, range (only if applicable), azimuth, and elevation.

Cellzcell outputs cell-by-cell information for each environment/mission in the order the environment/mission pairs are specified in the final results file. In the special case where the environment/mission queue in the final results file contains both "firepower, typical" and "mobility, typical" contexts, *cellzcell* automatically outputs an environment/mission column containing the maximum of the "mobility, typical" and "firepower, typical" values. This column will always be in the column after the latter of the two mission/environment pairs.

A view in a MUVES final results file is defined by the analysis approximation method (*e.g.*, "compart" for Compartment Model), the target-threat combination, and the shot pattern. *Cellzcell* only produces view output for the "compart" approximation method. A shot pattern may be a single shot, a group of single shots, or a pattern of shots (*e.g.*, grid, bivariate gaussian). Each shot is described by the azimuth, elevation, and impact location. Output for groups of shots is produced only if all shots in the group have a common azimuth and elevation.

By historical definition, cell-by-cell planar coordinates are the shot coordinates expressed as center of cell coordinates. For non-grid patterns, *e.g.*, a bivariate gaussian shot pattern, center of cell coordinates are not meaningful. For non-grid patterns, *cellzcell* always outputs the actual shot coordinates.

Options and their meanings are:

–a	Output the planar shot coordinates for grid shot patterns and groups of shots.
–p threat –r range	Display results for the specified *threat* at the specified *range* (*e.g.*, -p p9999 -r 1000). The capabilities of some threats are considered range-independent (up to a certain distance). In this case, the range option should not be used. However, if the target-threat range is important to a particular threat and the range is not specified, then output will be produced for all available ranges.
–t target	Display results for the specified *target* (*e.g.*, -t tank).
–u units	Display X,Y coordinates in the specified *units*, *e.g.*, -u meters. The specified units are also applied to the cell height and width options [-H cell_hgt and -V cell_wid]. Valid units are: feet (or ft), inches (or in), meters (or m), centimeters (or cm), and millimeters (or mm). If units are not specified, the default is millimeters.
–v view_no	Display results for the specified *view number* where the view number is an integer corresponding to a particular shot pattern. View number 0 corresponds to the first set of shots in the MUVES final results file, view number 1 corresponds to the second, *etc.*
–H cell_hgt	Specifies the cell height for binning shot locations (for a single shot or a group of single shots) to center of cell coordinates. Default is 100.
–W cell_wid	Specifies the cell width for binning shot locations (for a single shot or a group of single shots) to center of cell coordinates. Default is 100.

If options are not specified, *cellzcell* defaults to all "compart" views in the MUVES final results file. If an output file is not specified, *cellzcell* results are printed on standard output. If *cellzcell* is unsuccessful in its search, it prints an appropriate warning.

HINTS

For target, threat, or range options, the arguments must be specified exactly as they are in the MUVES final results file, otherwise, *cellzcell* is unsuccessful in its search.

SEE ALSO

MUVES Analyst's Guide

muves(1)

AUTHORS

Wendy A. Winner, BRL/VLD/VMB

Brian M. Rapp, BRL/VLD/VMB Student Contractor

NAME

dbchk – check MUVES region map file and MGED data base for consistency

SYNOPSIS

dbchk region_map_file MGED_file object[s]

DESCRIPTION

dbchk searches one or more objects in an MGED data base to determine whether all of its constituent regions are referenced by a MUVES region map file. Any regions which are not referenced will cause errors when encountered during a MUVES vulnerability analysis.

dbchk searches the entire tree for each object named on the command line. When it locates low level regions, it looks for them in the region map file, reporting the full path name of any regions which are not found. These should then be added to the region map file.

SEE ALSO

MUVES Analysts' Guide

Users Manual for BRL-CAD Graphics Editor MGED

AUTHOR

Jeff Hanes, BRL/VLD/VMB

NAME

gsbthrcv – MUVES-style threat inputs from VLD/GSB-style files

SYNOPSIS

gsbthrcv [-a author] [-c] [-d date] GSB_style_infile [MUVES_style_outfile]

DESCRIPTION

Gsbthrcv produces MUVES-style threat inputs from Vulnerability/Lethality Division (VLD) Ground Systems Branch (GSB) kinetic energy (KE) and shaped-charge (SC) threat files which were formatted for older versions of the Compartment Model. For KE rounds, MUVES "perf", "initial", and "range" files are output together. For SC projectiles, MUVES "pen", "phd", and "initial" files are output together. A question mark is placed in the output where MUVES required information cannot be extracted from the supplied GSB-style file. Users must manually replace question marks with correct information, must create separate threat files from this output, and should check the output produced.

Options and their meanings are:

–a author	Labels output with the specified author (*e.g.*, -a "John Smith").
–c	Produces commented output, *e.g.*, labels threat parameters, provides interpolation table headers, *etc.*
–d date	Labels output with specified date (*e.g.*, -d "90/08/29").

If an output file is not specified, *gsbthrcv* results are printed on standard out.

CAVEATS

No sorting is done on margin values of tabular data. Users should check margin values of interpolation tables to make certain these values are in strictly monotonic order.

SEE ALSO

MUVES Analyst's Guide

muves(1)

AUTHOR

Wendy A. Winner, BRL/VLD/VMB

NAME

ircompart – print "compart" method information from an intermediate results file.

SYNOPSIS

ircompart [**–i** item] [**–f** file] [**–u** units] intermediate_results_file

DESCRIPTION

ircompart reads a MUVES intermediate results file and prints requested information for every component in each shotline recorded in that file. The available selections are limited to those pertinent to the "compart" approximation method.

The information will be printed in the following form: For each shot, there will be a shot information line which shows the firing direction (both in azimuth and elevation and in vector form) and the shot location (in 2D view plane coordinates). Then there will be a single line of information for each component on the shot. The first item of information will be the threat type, which will be followed by the component name and any other requested items.

A shot information line might look like this:

az 30 el 0 dir < -.86601 -.5 0 > shot 0 100

A component information line will have the following form:

threat type component name [*requested item(s)*]

The threat type and component name will appear for every component. Other items may be requested using the syntax described below. Items will be separated by tabs and *all* requested information will be on one line per component. These lines can become quite long, so use discretion when selecting the items to be printed.

Options

–i *item name* — Read an item name. The recognized item names are described below. If there is only one item and no file specified (using the **–f** option), then the option letter may be omitted.

–f *file name* — Read a file of item names. The file may not contain more than one item name per line. The standard comment character (#) is recognized at the beginning of a line or on any line when it follows a tab.

–u *unit type* — Set output units for this run. Accepted abbreviations are ft, in, m, cm, mm.

Available Information

One may request basic geometric information about the components using the following items:

item name	*Description*
entry	Coordinates of entry point into component
normal	Normal vector to component surface at entry point
los	Component line of sight thickness
wtlos	Weighted line of sight thickness (for SC)
wtnorm	Weighted normal thickness (for KE)
nthick	Component normal thickness
obliq	Component obliquity

The information available about a threat depends on the type of threat used in the analysis. Specifically, a kinetic energy threat will have different parameters from a chemical energy (shaped charge) threat. Therefore, the information request must specify the threat type for the item of information desired. This is done using a prefix indicating the type of threat followed by a suffix indicating the desired threat parameter; the two are separated by a colon (e.g. "ke:avlpen" or "fp:stand"). Much of the information about shaped charge threats is further separated by the algorithm used to compute penetration. In order to select all of the information about a particular threat type, use an asterisk (*) for the suffix (e.g. "ke:*"). The available threat parameters are

listed below:

Standard kinetic energy threat

item nam	*Description*
ke:class	lass of KE projectile (see "compart" method documentation for explanation)
ke:projd	Projectile diameter in millimeters
ke:pend	Penetrator diameter in millimeters
ke:scale	Munition diameter scaling constant
ke:strike	Striking velocity
ke:vel	Velocity after threat has penetrated component
ke:ipen	Initial Penetration capability
ke:avlpen	Available Penetration capability

Standard shaped charge jet threat (used by both SC algorithms)

item name	*Description*
sc:wcd	Warhead charge diameter
sc:cutoff	Cutoff angle for warhead fuzing
sc:pitch	Pitch for canted warhead
sc:yaw	Yaw for canted warhead

Parameters for Fireman-Pugh algorithm

item name	*Description*
fp:stand	Distance to front of non-air component
fp:penred	Penetration reduction factor (initially 1.0)
fp:avlpen	Available penetration capability
fp:steqv	Total steel equivalent penetration
fp:tlos	Distance from munition to current component
fp:airlos	Contiguous airspace to current component
fp:airfx	Penetration reduction due to air space
fp:biso	Built-in standoff

Parameters for DSM algorithm

item name	*Description*
dsm:strike	Jet Striking Velocity
dsm:jdens	Jet material density
dsm:jdist	Jet propagation distance
dsm:minv	Minimum penetration velocity
dsm:btime	Jet breakup time
dsm:djetb	Diameter of jet at breakup time
dsm:exit	Exit time from previous component
dsm:speed	Jet penetration speed
dsm:steqv	Total steel equivalent penetration
dsm:depth	Penetration capability into current component
dsm:ratio	Ratio of RHA density to jet material density

Special parameters

item name	*Description*
range	Range to target (no prefix required) (currently used only for KE threats)
sand:angle	Obliquity of first sandwiched component (see "compart" method documentation for explanation)

There are a variety of values which may be used to compute the damage done to a particular component. All the values used by the "compart" method are listed below. Any or all of these may be chosen to track the damage done by the threat. Unlike the threat parameters, the damage parameters do not require a prefix; they are requested individually. The names are used as shown below.

Damage parameters

item name	*Description*
ammo	Boolean value indicating whether ammunition was hit by the main penetrator
phd	Profile Hole Diameter (for shaped-charge threat)
nfrags	Number of fragments (due to shaped-charge threat)
correc	Correction Factor (see "compart" method documentation for explanation)
elf	Expected lethal fragments
rhvol	Residual Hole Volume (for kinetic energy threat)
gtr	Gun tube ratio
holedm	Hole Diameter (due to kinetic energy threat)
track	Name of track portion hit by main penetrator (*i.e.* "edge" or "face")

Each damaged component contributes some value to the overall degradation of the target. In MUVES, these are called fractional remaining functionalities (FRF) and combined using positive logic operators. Historically, the inverse of these values has been used instead; this is called the loss of function (LOF). Either representation may be requested as output from "ircompart", depending on the form with which the analyst is most comfortable. To request the damage contributions for all components, give the type of representation desired ("frf" or "lof") followed by the environment/mission pair. The environment/mission pair must be separated by a colon(:). If a component was damaged, the value will be printed, otherwise a dash (-) will be printed in that column. The item names would look something like the following.

frf:environment:mission

lof:environment:mission

A star (*) in either field will match all possible values for that field. Thus,

frf:typical:*

will show FRFs for all missions for the "typical" environment, while

frf:*:*

will display all available environment/mission pairs.

Although this is similar to the regular expression wild card matching, it does not imply that full regular expressions are available. Only the star is recognized by this program.

(Unfortunately, it is not possible to display the system degradations, since they are combinations of the component degradations and are not explicitly recorded. If necessary, they can be recomputed by hand from the component values and the corresponding system definition file.)

AUTHOR

Jeff Hanes, BRL/VLD/VMB

NAME

irdump – print all contents of an intermediate results file.

SYNOPSIS

irdump [-G] [-H cell_hgt] [-W cell_wid] [-U units] intermediate_results_file

DESCRIPTION

irdump reads a MUVES intermediate results file and prints the information available for each shot in that file. Each shot in the file consists of several component traces, each of which will have some threat packet parameters and possibly some damage packets. *irdump* prints this information to the standard output in a general fashion; that is, it prints the contents of every packet without labeling the parameters. There is no interpretive or selective capability available with this post-processor.

Reading the output from *irdump* can be somewhat confusing. The user should consult the *MUVES Analysts' Guide* especially the section for the approximation method in use. This will describe the types and contents of all available packets.

The **–U** option allows the user to select the units in which the hit locations are printed. It *only* affects the shot locations and component hit points. Items within the packets are not converted to the specified units, since *irdump* has no way of knowing which items should be converted and which should not. The default is to print all values in millimeters.

The remaining options permit the analyst to print the cell coordinates of each shot (*i.e.* "grid" location) as well as the true coordinates of the shot. (These only differ when the shots are somehow randomly generated.) The **–G** option turns on this capability. If the shots were generated using a "grid" command, *irdump* will use the cell size specified by the analyst in the session file. For any other type of generated shots, the analyst may select the size of the desired grid using **–W** and **–H** to specify the cell width and height, respectively. This is useful when the original shot specifications are read from a view file, for instance. The default value for both parameters is 100 mm.

SEE ALSO

MUVES Analysts' Guide

AUTHOR

Jeff Hanes, BRL/VLD/VMB

NAME

muverat – perform one or more vulnerability analyses

SYNOPSIS

muverat [session_file]

DESCRIPTION

muverat is the vulnerability analysis program for the MUVES system. After reading the commands in a session file, it will load the requested files, raytrace the target, analyze the target's vulnerability to the threat(s), and store the results in a final results file.

muverat will read standard input if no session file is given on the command line.

CAVEAT

The MUVES user interface, ***muves***, facilitates input selection and maintains an audit trail of each analyst's work; ***muverat*** itself provides no audit trail maintenance capabilities. Therefore, the policy for using MUVES is that all production vulnerability analyses be done through the user interface in order to keep a record of the work performed at a given site.

Invoking ***muverat*** from the command line is allowed for experimental or developmental applications by those who prefer the command line interface and are willing to work without the added safeguards.

DIAGNOSTICS

There is a mind-boggling number of things that can go wrong. When an error is encountered, ***muverat*** will print an error message and exit with a 1 exit code. Because ***muverat*** exits immediately upon encountering an error, the results files will be incomplete in that event. Therefore, any results files it was creating should be removed.

SEE ALSO

muves(1)

MUVES Analysts' Guide

AUTHORS

VLD/VMB MUVES Development Team:

Doug Gwyn
Jeff Hanes
Scott Henry
Gary Moss
Karen Murray
Wendy Winner

NAME

muves – Modular UNIX®-based Vulnerability Estimation Suite

SYNOPSIS

muves

DESCRIPTION

MUVES is an integrated software system for performing vulnerability/lethality assessments. This system is designed to evaluate the interaction of a threat with a target where the target information is provided via ray-tracing geometric target descriptions. The MUVES user interface, *muves*, provides a menu-driven environment for specifying vulnerability analysis inputs, selecting parameters for a specific analysis, executing the specified analysis, and interpreting the analysis results.

When *muves* is invoked, the process fills the current window (or terminal screen on non-windowing systems) with a display that is divided into two parts; the top portion is devoted to a hierarchical "pop up" menu system; the bottom portion is dedicated to a scrolling text region. Between these portions, several status lines are reserved for prompted input, and for information of a transient nature (such as warning messages, informative messages about on-going computations, and help messages).

The menu hierarchy groups commands according to their function. The scrolling text region is used to display various types of information such as error messages, the contents of MUVES files, the status of analyses, or the output of postprocessors. Such information is stored internally so the user can view different portions of the scrolling region.

Commands and their default bindings for manipulating menus, for obtaining pop up menus, and editing menu prompts are listed in the following three tables. These bindings may be changed with a .muvesrc file in the user's home directory ($HOME/.muvesrc).

TABLE 1. Menu Manipulation Commands and their Default Bindings

Key	Command Name	Command Description
d	**next-menu-entry**	Move menu pointer down one entry.
u	**previous-menu-entry**	Move menu pointer up one entry.
space	**select-menu-entry**	Select highlighted menu entry.
h	**describe-menu-entry**	Get help about highlighted menu entry.
q	**quit-current-menu**	Quit current menu.

TABLE 2. Pop-up Menu Commands and their Default Bindings

Key	Command Name	Command Description
$	**set-up-options**	Tailor user interface.
@	**manage-analyses**	Get status/terminate analyses.
&	**enter-file-browser**	Examine MUVES data hierarchy.
?	**show-key-bindings**	Display menu of bound keys.

UNIX® is registered trademark of AT&T.

TABLE 3. Prompt Editing Commands

Key	Command Description
Delete	Delete the character behind the cursor.
Return	Enter response.
^A	Move the cursor to the beginning of the line.
^B	Move the cursor backward one character.
^D	Delete the character under the cursor.
^E	Move the cursor to the end of the line.
^F	Move the cursor forward one character.
^G	Abort the prompt without submitting a response.
^K	Erase all characters (kill) from the cursor to the end of the line.
^P	Insert the default response at the cursor location.
^R	Redraw the line.
^U	Erase all characters to the left of the cursor.
^V	Escape (insert) the next character typed.
^Y	Insert the contents of the kill buffer (see ^K) at the cursor.

MUVES maintains a UNIX file hierarchy which is accessible through the user interface and stores analysis input, session, log, intermediate results, and final results files. These files are grouped into directories called projects. Projects may only be created through the user interface.

Each MUVES analysis is referred to as a run. The series of runs made while a given project is being accessed is called a session. All selected analysis configuration parameters, including the names of input and output files, are automatically recorded during a session in a session file. All diagnostics from a session's analyses are stored in a log file. Final results files are created for each run. If information is needed on the intermediate computations of a MUVES calculation, selected internal parameters for a particular run may be requested. These parameters and values are stored in an intermediate results file.

All input files required to perform an analysis may be copied into a project via *muves*. Inputs for a specific analysis may be selected via menu entries or may be specified in response to prompts for keyboard input. *muves* automatically maintains a record of these selections and stores them for later reference in a session file. By maintaining a record of every input file used during an analysis, *muves* prevents accidental over-writing of input files which have been used for prior analyses. Once all parameters have been selected for a specific analysis, the analysis run is completed by selecting "analyze" menus. *muves* will automatically invoke *muverat* to perform the analysis computations.

Data from MUVES results files may also be interpreted or post-processed using *muves*. Data may be summarized or displayed graphically by making selections from the "results interpretation" menus; these post-processing tools may also be invoked from the command line.

ENVIRONMENT

All MUVES-related files are installed under a single directory. Users must set a MUVES environment variable equal to the absolute pathname of this root directory. It is recommended that users set this variable in in their automatic login shell configuration scripts. For example, if the standard UNIX Bourne shell is being used and the MUVES software is installed in /usr/muves, then *MUVES=/usr/muves export MUVES* should be entered into the user's .profile.

Users should also add $MUVES/bin to their command directory search path. For the standard UNIX Bourne shell, *PATH=$MUVES/bin:$PATH export PATH* will add $MUVES/bin to the command directory search path.

$MUVES/man should also be added to environmental variable specifying the manual page directory search path, for example, *MANPATH=$MUVES/man:$MANPATH export MANPATH.*

RESTRICTIONS

A set of tools are available for setting up selected input files. These tools are not accessible via the menu system and must be invoked from the command line.

Only the compartment-level model with lumped-parameter estimates for weapon system damage is implemented at this time. The design and implementation of the stochastic point-burst model under the MUVES environment is under development.

FILES

$HOME/.muvesrc – Commands read at the beginning of *muves* execution.

SEE ALSO

MUVES Analyst's Guide

MUVES Administrator's Guide

The Ballistic Research Laboratory CAD Package

Users Manual for BRL-CAD Graphics Editor

muverat(1), brlcad(1), cellxcell(1), colorsil(1), ircompart(1), irdump(1), siv(1)

DIAGNOSTICS

Error messages are intended to be self-explanatory.

SOURCE

VLD/VMB MUVES Development Team
Ballistic Research Laboratory
ATTN: SLBCR-VL-V
Aberdeen Proving Ground, MD 21005-5066

AUTHOR

Gary Moss

BUG REPORTS

Reports of bugs or problems should be submitted via electronic mail to <muves-bug@brl.army.mil>.

NAME

siv – convert MUVES final results to Compartment Model format

SYNOPSIS

siv [**–acis**] [**–d** interval] [**–f** file_root_name] [**–h** horiz_bias] [**–n** number] [**–p** threat] [**–q** target_code] [**–r** range] [**–u** units] [**–v** vert_bias] [**–w** view_info_file] [**–x** threat_code] final_results_file

DESCRIPTION

siv was created to allow comparison between results of MUVES and earlier vulnerability models (Compartment Model, Point Burst, etc.). It digests a MUVES final results file and writes the information in three different formats: the summary table, the view average tables, and the IUA (Individual Unit Action) file. The formats are are based on earlier versions of the Compartment Model; their formats are documented in the cited VAMF manual. Only the IUA format is rigorously controlled, since it is intended for use by FORTRAN programs. The others are for human consumption and are therefore somewhat more free-form. The desired output files are specified using the **-a** (for view average), **-i** (for IUA), and **-s** (for summary tables) options. Each of these options turns on that particular type of output. Thus, the command

siv -a -i

would specify both view average tables and IUA files, while excluding summary tables. If none of these options are specified, all three file types will be produced.

A MUVES final results file must be provided on the *siv* command line. It will usually be desirable to provide a view information file (described below) in order to get defilade values or non-uniform weighting of the views. The output is written to one of three files whose names are composed of the root file name (specified using the **–f** option) with an appropriate suffix appended. The suffixes are ".sum" for summary tables, ".avg" for view average files and ".iua" for IUA files. If no root file name is specified, the desired tables will be written to standard output.

The shots in the final results file must be in a MUVES "grid" shot pattern; any other pattern will be rejected by *siv*. The specifications for the grid are read from the MUVES final results file. A weight is computed for each cell and combined with the loss of function for the shot in that cell. These weighted values are combined to form the view averages. The weights are based on a bivariate Gaussian aim dispersion with equal horizontal and vertical standard deviations. A set of standard deviations is specified using the **-n** and **-d** options. **-n** sets the number of standard deviations to be used while **-d** sets the interval between them. Thus, the following arguments

-n 5 -d 300

would specify 5 circular standard deviations of 300, 600, 900, 1200, and 1500. Note that the interval should be given in the units desired for the output files (the above example would most likely be used for outputs reported in millimeters).

Options

–a This option specifies view average output which contains a set of tables of aim dispersions; one for each view.

–c By default, the weapon bias and the aim point (specified in the view information file) are offset from the target origin. This option allows the analyst to offset from the centroid of the view plane (*i.e.* the center of presented area).

–d *interval* This option is used to set the interval between standard deviations when weighting the cells. The default is 300, which assumes that the output files will be in millimeters. Analysts working in inches will most likely want to set this value to 12.

-f *file_root_name*	This option is used to set the root name for the output files. The desired outputs will be written to files with differing suffixes to distinguish between the various formats. If this option is not set, all tables will be written to standard output.
-h *horiz_bias*	This option allows the analyst to set a single horizontal bias to be applied throughout the *siv* run. This simulates horizontal aim error inherent in a particular weapon delivery system.
-i	This option specified IUA output, which is a table of weighted averages used by a variety of wargaming programs.
-n *number*	This option is used to change the number of standard deviations used for weighting the cells in each view. The default is 10.
-p *threat*	This allows the analyst to select from the threats listed in the final results file. If this option is not specified, *siv* will use the first threat listed in the file.
-q *target_code*	This option sets a 4 digit numeric code that will be used in the IUA file to identify the target used in this analysis. If this value is not set, the intended column in the IUA file will be left blank.
-r *range*	This option selects a threat range from among those available in the final results file (if any).
-s	This option specifies summary table output, which contains a table of weighted averages of all views in the final results file.
-u *units*	This option allows the analyst to set the output units for the view averaging information. MUVES final results files are always stored in terms of millimeters; therefore, this option must be set in order to convert the results to any other unit system. Linear measures used in other input files and in the option arguments (except the range) are expected to be in the units specified with this option (*i.e.* they are left unconverted).
-v *vert_bias*	This option allows the analyst to set a single vertical bias to be applied throughout the *siv* run. This simulates vertical aim error inherent in a particular weapon delivery system.
-w *view_info_file*	This is used to specify the path name of the view information file. If no such file is specified, the views will be weighted uniformly and only fully exposed conditions will be assessed.
-x *threat_code*	This option sets a 4 digit numeric code that will be used in the IUA file to identify the threat used in this analysis. If this value is not set, the intended column in the IUA file will be left blank.

View Information File

siv weights each shot in a MUVES final results file by computing an approximation of the Gaussian integral for the cell in which it falls. The view information file allows the analyst to specify the weight and center of aim for each view and to set the defilade conditions for that view.

In the view information line the analyst specifies a view by azimuth and elevation, a weight for that view, and an offset for the aim point. The weight is used in computing an overall average of the views. The aim point offset will be relative to the target origin or the centroid of the view; this can be changed using the -c option on the command line.

Following the view information line, the analyst may also specify one or more defilade conditions for averaging. Each must be specified on a separate line following the appropriate view line.

With each defilade condition, the analyst must specify the Y-value cutoff and the aim point offset for that defilade condition. Only those cells whose Y-values are greater than or equal to the cutoff value are included in the averaging. The analyst may specify as many defilade conditions as desired for each view. However, due to the current output formats, view information files should be written such that every view has the same number of defilade conditions and all defilades are in the same order.

Note that the units used for offsets and defilades in the view information file should be the same as desired in the output files (specified with the -u option).

A formal description of the view information file in Backus-Naur form follows:

```
file ::=          { view-spec }+

view-spec ::=     info-line { defl-line }*

info-line ::=     azim WS elev WS weight WS horiz-aim WS vert-aim NL
defl-line ::=     "defilade" WS cutoff WS horiz-aim WS vert-aim NL

cutoff ::=        real-value
azim ::=          real-value
elev ::=          real-value
weight ::=        real-value
horiz-aim ::=     real-value
vert-aim ::=      real-value

WS ::=            { <tab> or <space> }+
NL ::=            <new-line>

real-value ::=    <floating-point-number>
```

CAVEAT

Weights in the view information file are expected to add up to 1.0; however, *siv* presently does not check to insure that this is the case. Erroneous results may be produced if this is not checked.

EXAMPLE

A portion of a sample view information file is shown below. For a real analysis, this file would contain entries for more azimuths than shown (*e.g.* 60, 90, 120, 150, 180).

```
#    defilade and adjust values are in inches
#
#azim        elev        weight      xadj    yadj
#defilade    -           cutoff      xadj    yadj

0.0          0.0         0.279       0.00    -12.00
defilade                 4.0         0.00    18.00

30.0         0.0         0.201       12.00   -12.00
defilade                 4.0         12.00   18.00
```

If this view information file were titled "view_info", one could use it with the following invocation:

```
siv -f out -w view_info -u inches 42.fr
```

Assuming there was also a final results file named "42.fr". The tabular results would appear in three files called "out.sum", "out.iua", and "out.avg".

SEE ALSO

Nail, C. L., Bearden, T. E., and Jackson, E., "Vulnerability Analysis Methodology Program (VAMP): A Combined Compartment-Kill Vulnerability Model," Computer Sciences Corporation

Technical Manual, CSC TR-79-5585, October 1979.

Brooks, Wilbert J., and Johnson, W. Donald, "Horizontal Attack Angle Distribution for Armored Vehicles", US Army Materiel Systema Analysis Activity, Technical Report No. 481, November 1989.

MUVES Analysts' Guide

muves(1)

AUTHOR

Jeff Hanes, BRL/VLD/VMB

BUGS

If *siv* is given a file that is not a MUVES final results file, it will dump core rather than giving a warning message and exiting gracefully.

INTENTIONALLY LEFT BLANK.

Glossary

analysis — An "analysis" consists of the process of computing functionalities using a single approximation method for a single target interacting with one or more threats traveling along paths defined by one or more shot groups.

approximation method — The collection of assumptions, simplifications, empirical data, and mathematical models used to approximate the physical processes involved in the interaction of the threat with the target. Within MUVES, these are implemented in a set of interaction modules and evaluation modules which determine the damage to each target component in the course of an analysis. The "compart" approximation method is the first method implemented in MUVES.

component — A component is a piece of a target with a distinct identity, with which threats may interact. For example, in the compartment model, the crew compartment is modeled as a single component; whereas, in a point-burst model, it would be modeled as many items, each having its own properties and playing its own role in the overall functionality of the target. For targets modeled using MGED (the only geometric method currently supported), one or more regions are assigned to each target component by an analyst-provided "region map" file.

component category — A grouping of components for which threat interactions and damage are computed in an identical fashion. Note that this does not imply that these computations must use the same data, only that the same *type* of data must be available and will be used in the same way for all components in the same category.

component trace — A component trace is the geometric information calculated for a single component during the course of a ray-trace through the target. Depending on the needs of the approximation method, it may contain entry and exit points, normal vectors, and curvature information. This information is available to the analyst via the optional intermediate results file.

context — This is the name given to the combination of environment and mission function. It is the set of circumstances and operations under which the analyst wishes to determine the remaining functionality of the target.

critical system — A system is critical if it is contained in the damage assessment expression for a context specified in the current analysis session.

critical component — 1. A critical component is any component which, if damaged, would result in the reduction of the target's ability to perform a mission function (i.e. mobility capability).
2. A critical component is any component specified in the damage assessment expression for a context in the current analysis session, either directly or through a system definition.

damage assessment expression — A damage assessment expression defines the contributions various systems make to the functionality of the vehicle for a particular context. These contributions are defined in the form of logical and mathematical combinations, using the same grammar as system definitions. This grammar provides the functionality found in existing vulnerability codes while allowing much greater flexibility.

damage evaluation selection — A given type of damage may be evaluated in more than one manner. Also, different types of damage require different forms of evaluation. Each type of evaluation is isolated in a single module within MUVES. Therefore, it is necessary to specify the desired module for each category of component that may be hit in a given analysis. Every analysis *must* specify one of these files. Normally, there will be a standard file for a given target. Unless the analyst wishes to do an unusual form of evaluation, it may not be necessary to change this file.

damage packets — A data structure which defines the type of damage done to a component and contains any parameters needed to describe the degree of damage (*e.g.* number of impacting fragments, hole diameter, deposited energy). Note that this data structure only contains the physical parameters of the damage and contains no information about the effect of that damage on the component's functionality.

environment — The environment refers to the physical and circumstantial surroundings of the target (*e.g.*, "typical", "european, offensive", "middle east", or "driving down the road in heavy fog").

evaluation module — This is a module written by the approximation method developer to compute the functionality of a component based on the damage to that component during its interaction(s) with the threat(s). MUVES invokes these modules based on the component category and the specifications in the damage evaluation selection file. Each evaluation module specifies the type of damage that is relevant to the calculations it must perform; the interaction modules will only produce damage that has been declared relevant to a particular component category.

final results — The final results are the remaining functionalities for the target in specified contexts after interaction with a threat. The final results file contains queueing information to help differentiate between the various views (there may be many in a given final results file), followed by all of the shot results for the session. Each line of the final results file shows the origin and direction for a

single initial shot and the resulting target functionalities in all of the requested contexts.

functionality/utility — Vulnerability results have traditionally been expressed in terms of "probability of kill" or "loss of function" for a given target. There has been a great deal of discussion concerning the exact meaning of these terms which will not be repeated here. The MUVES philosophy has been to refer to the final results values as "utilities" or "functionalities". The exact meaning of these numbers is left to the approximation method designer, but in general they are used to define the probability of the target or subsystem retaining total functionality. The primary reason for this is that it allows the analyst to write his deactivation diagrams and damage assessment expressions in terms of positive combinations rather than negative; this makes the expressions easier to write and thus less error prone. For certain methods, however, the interpretation of these values may differ significantly; one should always carefully read the approximation method documentation.

initial threat path — A threat path is a list of component traces from a ray-traced path through a target plus the parameters for a threat impacting the first component (*before* any interactions have occurred).

interaction module — This is a function written by the approximation method developer to perform the calculations involved in the interaction of a single component with a threat. MUVES selects these functions based on the approximation method, the component category, and the threat type. Therefore, components must be assigned to appropriate categories based on the way they are treated when interacting with threats. Detailed information about available component categories and threat types is contained in the appendix for a particular approximation method. Each interaction module must be able to deal with all of the possible outcomes of that interaction. For instance, an armor interaction module in a point-burst model must be able to create the threat paths for a spall cloud if it determines that the armor is perforated. If the method performed blast calculations along with fragment vulnerability, the interaction module would also have to be able to produce a shock wave in an appropriate manner.

intermediate results — The analyst may specify that intermediate results be stored for a given analysis. MUVES will store all geometric, component, and threat information for a given analysis in a user specified file. These files can become quite large; therefore, analysts should write intermediate results only when necessary and remove the files as soon as possible. The primary use for this information is to give the analyst insight into the internal computations performed during an analysis. This can improve his understanding of the processes involved, or be used to debug a target or threat when anomalies are discovered in the final results.

mission function — A mission function is the operation the target is expected to perform. Initially, the mission functions used in MUVES will generally correspond to the kill criteria used in traditional

vulnerability codes (*e.g.*, mobility, firepower, catastrophic, *etc.*).

session file — The parameters for each analysis are recorded in a "session file" for that analysis. This permits an analyst to examine previous runs, compare results, or setup a new run with similar parameters. Archiving session files with the corresponding results and requisite input files produces an official record of the analysis for future reference. Once a session file is created by the user interface, it will never be altered; that would invalidate any later attempts to archive that session. It is possible to run a variation of a previous analysis by loading the corresponding session file and changing the desired parameters.

shot — A shot is a point and a vector which collectively specify an initial ray-trace operation. The point specifies the origin of the trace in the target coordinate system, while the vector specifies the desired direction.

shot group — A "shot group" is a collection of one or more shots, each of which may have a shot pattern attached; within MUVES, this is synonymous with the word "view". Shot groups may be specified explicitly using the "group" keyword, followed by several shots (with possible shot patterns) and ending with the "endgroup" keyword. Thus, one could combine several small grids covering specific portions of the target into a single view. If necessary, one may specify any desired arrangement of shots by repeatedly specifying single shots in the same shot group. If a particular shot specification is not enclosed by the "group" and "endgroup" keywords, then the shot (or the shots generated from its shot pattern) is treated as a separate shot group.

shot pattern — A shot pattern specifies a set of shots. A pattern may consist of a single shot, or many shots clustered around an orienting shot. There are two standard patterns currently provided in MUVES, though others may be added later if necessary. The most commonly used pattern is the "grid" pattern which locates the shot origins in a regular array on a plane containing the orienting shot, while their directions are parallel to the orienting shot. MUVES also provides the "gauss" pattern wherein the shot origins are randomly distributed in a bivariate Gaussian distribution in a plane; otherwise, it is similar to the "grid" pattern.

system — A system is any combination of components or subsystems using logical and/or mathematical operators. For instance, the fuel system might consist of the fuel tank, fuel, fuel pump, fuel lines, and carburetor. All of these would be combined using 'and' operators to indicate that all of them are necessary for proper functioning of the system.

system definitions — System definitions are specified in a file containing the definitions for all of the systems in the target. Each definition has the form: <system> = <expression>, where

<expression> is a logical and/or mathematical combination of components or subsystems. The meaning of the definition is: "for the named system to be completely functional, the following combination of components and subsystems must be completely functional". For approximation methods that support fractional functionality, the logical conditions are generalized in accordance to the laws of probability.

system degradation — When components within a system are damaged, the system is said to be degraded in that it can no longer perform its intended function at full capacity. Degradation is a measure of the loss in capability for the system in question. Numerically, this is the inverse of the remaining utility of the system (assuming that utility is measured from 0 to 1).

target — A target is a "system of systems", having definite geometry and designed to perform some mission or missions of interest. Components form the lowest level of target structure; they are the "building blocks" of the system hierarchy.

target characteristics — All information needed to characterize a specific target is kept in files located in project inputs directories. This includes files containing component definitions, component category assignments, and component properties as well as system definitions and damage assessment expressions for contexts of interest.

target geometry — This refers to a file containing the geometric description of the target. That is, it gives the size, shape, and position of each component in the target. This description is interrogated by the ray tracer during the course of a MUVES analysis.

threat — A threat is any phenomenon that can produce some kind of damage to a target. Note that both the phenomenon and the damage can be rather abstract (*e.g.* line of sight, detection) if desired.

threat path — A list of component traces from a ray-traced path through a target plus the parameters for a threat propagating along that path. This information is required to propagate a threat through the series of interaction modules in order to compute the damage to the target as a result of threat/component interactions; the subsequent components in a threat path may also change as a result of the interactions.

threat type — A category of threats which behave in the same fashion when interacting with target components (*e.g.* shaped charge jet, spall fragment).

view — Any set of shots used to interrogate a target. (*see also* "shot group")

List of Symbols and Abbreviations

ammo	-	ammunition
AP	-	Armor-Piercing
ap	-	armor-piercing
APC	-	Armored Personnel Carrier
APG	-	Aberdeen Proving Ground
ASCII	-	American Standard Code for Information Interchange
AVG	-	average
avg	-	average
az	-	azimuth
azel	-	azimuth/elevation
azim	-	azimuth
BISO	-	Built-in Standoff
BNF	-	Backus-Naur Format
BRL	-	Ballistic Research Laboratory
BRLCAD	-	Ballistic Research Laboratory Computer-Aided Design package
CAD	-	Computer-Aided Design
ccmap	-	component category map (file)
CE	-	Chemical Energy
cellxcell	-	Compartment-style cell-by-cell postprocessor
cm	-	centimeters
colorsil	-	Color silhouette postprocessor
comp	-	component identifier map (file)
compart	-	Compartment approximation method
CPU	-	Central Processing Unit
CSC	-	Computer Sciences Corporation
dae	-	damage assessment expression (file)
deg	-	degree
des	-	damage evaluation selections (file)
diam	-	diameter
DMD	-	Dot-Mapped Display
DSM	-	DiPersio, Simon, Merendino
ecurve	-	evaluation curves for EMs (file)
EFP	-	Explosively Formed Projectiles
elev	-	elevation
EM	-	Evaluation Module
Esc	-	escape key

F	-	Firepower
FORAST	-	programming language
FP	-	Fireman-Pugh
fr	-	final results
FRF	-	Fractional Remaining Functionality
ft	-	foot or feet
geom	-	target database description (file)
GEOM	-	threat packet
GIFT	-	Geometric Information for Targets
gps	-	Gunner's Primary Sight
GSB	-	Ground Systems Branch of the BRL/VLD
horiz	-	horizontal
HVAP	-	Hyper Velocity Armor Piercing Projectiles
icurve	-	interaction curves for IMs (file)
IDs	-	identifiers
IM	-	Interaction Module
in	-	inches
IUA	-	Individual Unit Action
IV	-	Internal Volume
IVP	-	Internal Volume Perforated
K	-	Kill
KE	-	Kinetic Energy
ke	-	kinetic energy
LOF	-	Loss of Function
LRP	-	Long Rod Penetrator
lt	-	light
m	-	meters
M	-	Mobility
mm	-	millimeters
MF	-	Mobility or Firepower
MGED	-	Multi-device Graphics EDitor
MNP	-	mobility kill with no perforation
MOF	-	Mobility or Firepower
MUVES	-	Modular UNIX-based Vulnerability Estimation Suite
muverat	-	muverat executable image
muves	-	muves executable image
NL	-	newline
obl	-	obliquity
pen	-	penetration

perf	-	perforation
phd	-	profile hole diameter
PK	-	probability of kill
pkt	-	packet
prop	-	component properties (file)
RHA	-	Rolled Homogeneous Armor
RHV	-	Residual Hole Volume
rsh	-	remote shell
rt	-	right
SC	-	Shaped-Charge
sc	-	shaped-charge
SCCS	-	Source Code Control System
SCJ	-	Shaped-Charge Jet
SCJ_DSM	-	Shaped-Charge Jet threat packet using DSM parameters
SCJ_FP	-	Shaped-Charge Jet threat packet using Fireman-Pugh paramaters
SCJ_STD	-	Standard Shaped-Charge Jet threat packet
SCW	-	Shaped-Charge Warhead
SECAD	-	System Engineering and Concepts Analysis Division (of the BRL)
siv	-	Summary tables, IUA files and View averages postprocessor
SLAVE	-	Simple Lethality and Vulnerability Estimator
SQuASH	-	Stochastic Qualitative Analysis of System Hierarchies
STD	-	standard
sysdef	-	system definitions (file)
TGT	-	target
Ui	-	User Interface
US	-	United States
VAMP	-	Vulnerability Analysis Methodology Program
VAST	-	Vulnerability Analysis for Surface Targets
VC	-	Vehicle Code
vert	-	vertical
VLD	-	Vulnerability/Lethality Division (of the BRL)
VMB	-	Vulnerability Methodology Branch (of the BRL/VLD)
vol	-	volume
WCD	-	Warhead Charge Diameter
wtd	-	weighted

INTENTIONALLY LEFT BLANK.

No. of Copies	Organization
2	Administrator Defense Technical Info Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22304-6145
1	Commander U.S. Army Materiel Command ATTN: AMCAM 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander U.S. Army Laboratory Command ATTN: AMSLC-DL 2800 Powder Mill Road Adelphi, MD 20783-1145
2	Commander U.S. Army Armament Research, Development, and Engineering Center ATTN: SMCAR-IMI-I Picatinny Arsenal, NJ 07806-5000
2	Commander U.S. Army Armament Research, Development, and Engineering Center ATTN: SMCAR-TDC Picatinny Arsenal, NJ 07806-5000
1	Director Benet Weapons Laboratory U.S. Army Armament Research, Development, and Engineering Center ATTN: SMCAR-CCB-TL Watervliet, NY 12189-4050
(Unclass. only) 1	Commander U.S. Army Armament, Munitions and Chemical Command ATTN: AMSMC-IMF-L Rock Island, IL 61299-5000
1	Director U.S. Army Aviation Research and Technology Activity ATTN: SAVRT-R (Library) M/S 219-3 Ames Research Center Moffett Field, CA 94035-1000

No. of Copies	Organization
1	Commander U.S. Army Missile Command ATTN: AMSMI-RD-CS-R (DOC) Redstone Arsenal, AL 35898-5010
1	Commander U.S. Army Tank-Automotive Command ATTN: ASQNC-TAC-DIT (Technical Information Center) Warren, MI 48397-5000
1	Director U.S. Army TRADOC Analysis Command ATTN: ATRC-WSR White Sands Missile Range, NM 88002-5502
1	Commandant U.S. Army Field Artillery School ATTN: ATSF-CSI Ft. Sill, OK 73503-5000
(Class. only) 1	Commandant U.S. Army Infantry School ATTN: ATSH-CD (Security Mgr.) Fort Benning, GA 31905-5660
(Unclass. only) 1	Commandant U.S. Army Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905-5660
1	Air Force Armament Laboratory ATTN: WL/MNOI Eglin AFB, FL 32542-5000
	Aberdeen Proving Ground
2	Dir, USAMSAA ATTN: AMXSY-D AMXSY-MP, H. Cohen
1	Cdr, USATECOM ATTN: AMSTE-TC
3	Cdr, CRDEC, AMCCOM ATTN: SMCCR-RSP-A SMCCR-MU SMCCR-MSI
1	Dir, VLAMO ATTN: AMSLC-VL-D
10	Dir, BRL ATTN: SLCBR-DD-T

No. of Copies	Organization
10	C.I.A. OIR/DB/Standard GE47 HQ Washington, DC 20505
1	HQDA DAMI-FIT (COL Everson) WASH DC 20310-1001
1	HQDA DAMO-ZD (Mr. Riente) The Pentagon, Rm 3A538 WASH DC 20310-0410
1	HQDA SARD-DO (Mr. Ron Mlinarchik) WASH DC 20310-0102
1	HQDA SARD-DOV (COL Steve Dasher) WASH DC 20310-0103
1	HQDA SARD-TN (LTC Fejfar) The Pentagon, Rm 3E360 WASH DC 20310
1	HQDA SARD-TT (Mr. Mike Clauson) WASH DC 20310-0103
1	HQDA OUSDA(A), DDRE(T&E) ATTN: COL Bernard (Chip) B. Ferguson The Pentagon WASH DC 20301-3110
2	HQDA OUSD(A) DDDRE(R&AT-ET) ATTN: Mr. Rick L. Menz/Dr. James Short The Pentagon, Rm 3D1089 WASH DC 20301-3080
1	HQDA (Limres Study Group) ATTN: Shirley D. Ford The Pentagon, Room 1B929 WASH DC 20310
1	Office of the Assistant Secretary of the Army (Research, Development, and Acquisition) ATTN: LTG Cianciolo, Military Deputy Washington, DC 20310-0100
1	Office of the Secretary of the Army (Research, Development, and Acquisition) ATTN: MG Belt.son, Deputy for Systems Management Washington, DC 20310-0103
1	Deputy Under Secretary of the Army for Operations Research ATTN: SAUS-OR (Hon Walt Hollis) The Pentagon, Room 2E660 Washington, DC 20310-0102
1	Office of the Deputy Director of Defense, R&E ATTN: Dr. William Snowden The Pentagon, Room 3D359 Washington, DC 20301
1	Office of the Asst Dep Dir of Defense Live Fire Testing ATTN: COL L. Stanford The Pentagon, Room 3E1060 Washington, DC 20301
2	OSD OUSD (A) ODDDRE (T&E/LFT) ATTN: James O'Bryon Albert E. Rainis The Pentagon, Rm 3E1060 Washington, DC 20301-3110
1	SAF/AQT (Mr. George Warren) The Pentagon, Rm BE939 Washington, DC 20330-1000
1	Assistant Deputy Under Secretary of the Navy ATTN: Fred Crowson Crystal Plaza 5, Room 162 2211 Jefferson Davis Hwy. Arlington, VA 22202

No. of Copies	Organization
9	Defense Advanced Research Projects Agency ATTN: Mr. B. Bandy Dr. R. Kahn Dr. C. Kelly Mr. P. Losleben Dr. J. Lupo Mr. F. Patten Dr. Reynolds Mr. S. Squires COL J. Thorpe 1400 Wilson Boulevard Arlington, VA 22209
2	Central Intelligence Agency ATTN: ORD/PERD (Ray Cwiklinski) (Tom Kennedy) Washington, DC 20505
1	Central Intelligence Agency ATTN: ORD/IERD (J. Fleisher) Washington, DC 20505
2	Central Intelligence Agency ATTN: OIA (Barbara A. Kroggel) (Monica McGuinn) Washington, DC 20505
1	Central Intelligence Agency ATTN: ORD (Donald Gerson) 1820 N. Fort Myer Drive Arlington, VA 22209
1	Chief of Naval Operations OP-03-C2 ATTN: CPT P.X. Rinn Rm 4D537, The Pentagon Washington, DC 20350-2000
1	Department of the Navy ATTN: RADM Charles R. McGrail, Jr. Pentagon, Room 4E536 Washington, DC 20350-2000
1	Commander U.S. Army Materiel Command ATTN: AMCDE-PI (Dan Marks) 5001 Eisenhower Avenue Alexandria, VA 22333-0001

No. of Copies	Organization
1	Commander U.S. Army Materiel Command ATTN: AMCSI (Dr. R. Chait) 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander U.S. Army Materiel Command ATTN: AMCPD (Darold Griffin) 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander U.S. Army Materiel Command ATTN: AMCPD-PM (Jim Sullivan) 5001 Eisenhower Avenue Alexandria, VA 22333-0001
2	Commander U.S. Army Materiel Command ATTN: AMCPM-LOTA (Robert Hall) (MAJ Purdin) 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander U.S. Army Materiel Command ATTN: AMCPD-PT (Alan Elkins) 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander U.S. Army Laboratory Command ATTN: AMSLC-CT (K. Zastrow) 2800 Powder Mill Road Adelphi, MD 20783-1145
1	Commander U.S. Army Laboratory Command ATTN: AMSLC-CG 2800 Powder Mill Road Adelphi, MD 20783-1145
1	Commander U.S. Army Laboratory Command ATTN: AMSLC-LO (LTC P. J. Fardink) 2800 Powder Mill Road Adelphi, MD 20783-1145

No. of Copies	Organization
1	Commander U.S. Army Laboratory Command ATTN: SLCLT (LTC Marshall) 2800 Powder Mill Road Adelphi, MD 20783-1145
2	Commander U.S. Army Laboratory Command ATTN: AMSLC-TP (J. Predham) (D. Smith) 2800 Powder Mill Road Adelphi, MD 20783-1145
1	Commander U.S. Army Laboratory Command ATTN: SLCTO (Marcos Sola) 2800 Powder Mill Road Adelphi, MD 20783-1145
1	Commandant U.S. Army Logistics Management College ATTN: AMXMC-LS-S (CPT(P) Stephen Parker) Ft. Lee, VA 23801
1	Director Combat Development U.S. Army Transportation School ATTN: COL Elijah Toney Ft. Eustis, VA 23604
2	Commander U.S. Army Materials Technology Laboratory ATTN: SLCMT-ATL SLCMT-MRD (Dr. Robert Heyman) Watertown, MA 02172-0001
3	Director U.S. Army Research Office ATTN: SLCRO-MA (Dr. J. Chandra) (Dr. K. Clark) (Dr. Wu) P.O. Box 12211 Research Triangle Park, NC 27709-2211

No. of Copies	Organization
1	Director U.S. Army Survivability Management Office ATTN: SLCSM-C31 (H. J. Davis) 2800 Powder Mill Road Adelphi, MD 20783
1	Director U.S. Army Survivability Management Office ATTN: SLCSM-D (COL H. Head) 2800 Powder Mill Road Adelphi, MD 20783-1145
1	Commander U.S. Army Armament Research, Development and Engineering Center ATTN: SMCAR-CCH-V (Paul H. Gemmill) Picatinny Arsenal, NJ 07806-5000
2	Commander U.S. Army Armament Research, Development and Engineering Center ATTN: SMCAR-FSS-E (Jack Brooks) (Edward B. Lacher) Picatinny Arsenal, NJ 07806-5000
1	Commander U.S. Army Armament Research, Development and Engineering Center ATTN: SMCAR-TD (Jim Killen) Picatinny Arsenal, NJ 07806-5000
1	Commander U.S. Army Armament, Research, Development and Engineering Center ATTN: SMCAR-TDS (Vic Lindner) Picatinny Arsenal, NJ 07806-5000
1	Commander Belvoir Research, Development and Engineering Center ATTN: STRBE-FC (Ash Patil) Fort Belvoir, VA 22060-5606
1	Commander Belvoir Research, Development and Engineering Center ATTN: STRBE-JDA (Melvin Goss) Fort Belvoir, VA 22060-5606

No. of Copies	Organization
1	Commander, USACECOM R&D Technical Library ATTN: ASQNC-ELC-IS-L-R, Myer Center Fort Monmouth, NJ 07703-5000
1	Director Center for Night Vision and Electro-Optics ATTN: AMSEL-RD-NV-V (John Palmer) Fort Belvoir, VA 22060-5677
1	Director Center for Night Vision and Electro-Optics ATTN: AMSEL-RD-NV-V (John Ho) Fort Belvoir, VA 22060-5677
1	Director Center for Night Vision and Electro-Optics ATTN: AMSEL-RD-NV-D (Dr. R. Buser) Fort Belvoir, VA 22060-5677
1	Commander U.S. Army Foreign Science and Technology Center ATTN: AIFR (Bill Rich) 220 Seventh Street, NE Charlottesville, VA 22901-5396
4	Commander U.S. Army Foreign Science and Technology Center ATTN: AIFRS (T. Walker) (D. Hardin) (R. Wittnebel) (John Aker) 220 Seventh Street, NE Charlottesville, VA 22901-5396
2	Commander U.S. Army Foreign Science and Technology Center ATTN: AIFRS (Gordon Spencer) (Dr. Steven Carter) 220 Seventh Street, NE Charlottesville, VA 22901-5396
1	Commander U.S. Army Foreign Science and Technology Center ATTN: AIFRT (John Kosiewicz) 220 Seventh Street, NE Charlottesville, VA 22901-5396
1	Commander U.S. Army Foreign Science and Technology Center ATTN: AIFRE (S. Eitelman) 220 Seventh Street, NE Charlottesville, VA 22901-5396
1	Commander U.S. Army Harry Diamond Laboratories ATTN: SLCHD-RT (Peter Johnson) 2800 Powder Mill Road Adelphi, MD 20783-1197
1	Commander U.S. Army INSCOM ATTN: IAOPS-SE-M (George Maxfield) Arlington Hall Station Arlington, VA 22212-5000
2	Commander U.S. Army Missile Command ATTN: AMSMI-RD-GC-T (R. Alongi) Redstone Arsenal, AL 35898-5000
1	Commander US Army Missile Command ATTN: AMSMI-RD-SS-AA (Kim Cornelius) Redstone Arsenal, AL 35898-5252
1	Commander U.S. Army Missile Command ATTN: AMSMI-RD-SS-AT (Ed Vaughn) Redstone Arsenal, AL 35898-5000
1	Commander US Army Missile Command ATTN: AMSMI-RD-ST-WF (Lynn S. Craft) Redstone Arsenal, AL 35898-5247

No. of Copies	Organization
1	Commander U.S. Army Missile Command ATTN: AMSMI-RD (J. Bradas) Redstone Arsenal, AL 35898-5000
1	Commander U.S. Army Missile Command ATTN: AMSMI-YTSD (Glenn Allison) Redstone Arsenal, AL 35898-5070
1	Commander U.S. Army Missile Command ATTN: AMSMI-REX (W. Pittman) Redstone Arsenal, AL 35898-5500
1	Director U.S. Army Missile and Space Intelligence Center ATTN: AIMS-RT (Pat Jordan) Redstone Arsenal, AL 35898-5500
1	Director U.S. Army Missile and Space Intelligence Center ATTN: AIMS-YLD (Vernon L. Stallcup) Redstone Arsenal, AL 35898-5500
2	Director U.S. Army Missile and Space Intelligence Center ATTN: AIMS-YRS (Thomas Blalock) (Pete Kirkland) Redstone Arsenal, AL 35898-5500
2	Director U.S. Army Missile and Space Intelligence Center ATTN: AIMS-YRT (Francis G. Cline) (Don A. Slaymaker) Redstone Arsenal, AL 35898-5500
1	Director U.S. Army Missile and Space Intelligence Center ATTN: Randy L. Smith Redstone Arsenal, AL 35898-5500

No. of Copies	Organization
1	Commander U.S. Army Natick R&D Center ATTN: STRNC-OI (Stephen A. Freitas) Natick, MA 01760
1	Commander U.S. Army Tank-Automotive Command ATTN: AMCPM-BLK-III (COL Don Derrah) Warren, MI 48397-5000
1	Commander U.S. Army Tank-Automotive Command ATTN: AMSTA-CK (M. Erickson) Warren, MI 48090-5000
1	Commander U.S. Army Tank-Automotive Command ATTN: AMSTA-CK (Newell) Warren, MI 48090-5000
1	Commander U.S. Army Tank-Automotive Command ATTN: AMSTA-CR (Mr. Wheelock) Warren, MI 48397-5000
1	Commander U.S. Army Tank-Automotive Command ATTN: AMSTA-CV (COL Becking) Warren, MI 48397-5000
2	Commander U.S. Army Tank-Automotive Command ATTN: AMSTA-NKS (D. Cyaye) (J. Rowe) Warren, MI 48397-5000
2	Commander U.S. Army Tank-Automotive Command ATTN: AMSTA-RG (R. Munt) (R. McClelland) Warren, MI 48397-5000
2	Commander U.S. Army Tank-Automotive Command ATTN: AMSTA-RSC (John Bennett) (Wally Mick) Warren, MI 48397-5000

No. of Copies | Organization

1 Commander
U.S. Army Tank-Automotive Command
ATTN: AMSTA-RSK (Sam Goodman)
Warren, MI 48090-5000

1 Commander
U.S. Army Tank-Automotive Command
ATTN: AMSTA-RY (Ron Beck)
Warren, MI 48090-5000

7 Commander
U.S. Army Tank-Automotive Command
ATTN: AMSTA-ZE (R. Asoklis)
AMSTA-ZEA (C. Robinson)
(R. Gonzalez)
(A. Iverson)
AMSTA-ZS (D. Rees)
AMSTA-ZSS (J. Thompson)
(J. Soltez)
Warren, MI 48397-5000

1 Office of the PEO, Armored Sys Mod
ATTN: SFAE-ASM-CV (Brian Bonkosky)
Warren, MI 48397-5000

1 Commander
HQ, TRADOC
ATTN: Asst Dep Chief of Staff
for Combat Operations
Fort Monroe, VA 23651-5000

2 Director
HQ, TRAC RPD
ATTN: ATRC-RP (COL Brinkley)
ATRC-RPR (Mark W. Murray)
Ft. Monroe, VA 23651-5143

1 Director
U.S. Army Cold Regions Research and
Development Laboratory
ATTN: Technical Director (Lewis Link)
72 Lyme Road
Hanover, NH 03755

No. of Copies | Organization

1 U.S. Army Corps of Engineers
Assistant Director Research and Development
Directorate
ATTN: Mr. B. Benn
20 Massachusetts Avenue, NW
Washington, DC 20314-1000

1 Commander
U.S. Army Operational Test and Evaluation
Agency
ATTN: MG Stephenson
4501 Ford Avenue
Alexandria, VA 22302-1458

1 Commander
U.S. Army Vulnerability Assessment
Laboratory
ATTN: SLCVA-CF (Gil Apodaca)
White Sands Missile Range, NM 88002-5513

1 Director
TRAC-WSMR
ATTN: ATRC-RD (McCoy)
WSMR, NM 88002-5502

2 U.S. General Accounting Office
Program Evaluation and Methodology
Division
ATTN: Robert G. Orwin
Joseph Sonnefeld
Room 5844
441 G Street, NW
Washington, DC 20548

1 Director
U.S. Army Model Improvement and Study
Management Agency
ATTN: SFUS-MIS (Eugene P. Visco)
1900 Half Street, SW, Rm L101
Washington, DC 20324

1 Director
U.S. Army Industrial Base Engineering
Activity
ATTN: AMXIB-MT
Rock Island, IL 61299-7260

No. of Copies	Organization
1	Director U.S. Army Industrial Base Engineering Activity ATTN: AMXIB-PS (Steve McGlone) Rock Island, IL 61299-7260
3	Director U.S. Army Engineer Waterways Experiment Station ATTN: WESEN (Dr. V. LaGarde) (Mr. W. Grabau) WESEN-C (Mr. David Meeker) PO Box 631 Vicksburg, MS 39180-0631
1	U.S. Army Engineer Topographic Laboratories ATTN: Technical Director (W. Boge) Fort Belvoir, VA 22060-5546
1	Commander U.S. Army Operational Test and Evaluation Agency ATTN: LTC Gordon Crupper 4501 Ford Ave. #870 Alexandria, VA 22302-1435
1	Commander David Taylor Research Center Code 1702 (Mr. Robert Wunderlick) Bethesda, MD 20084-5000
1	Commander David Taylor Research Center Code 1740.2 (Mr. Fred J. Fisch) Bethesda, MD 20084-5000
1	Commander David Taylor Research Center Code 1750 (Mr. William Conley) Bethesda, MD 20084-5000
1	Director Lawrence Livermore National Laboratory ATTN: Mark Wilkins (L-3321) PO Box 808 Livermore, CA 94551

No. of Copies	Organization
3	Director Los Alamos National Laboratory ATTN: Dean C. Nelson, MS 985 Gary Tietgen, MS F600 Terrence Phillips, MS G787 PO Box 1663 Los Alamos, NM 87545
1	Director Los Alamos National Laboratory ATTN: LTC Michael V. Ziehmn, MS F681 USMC PO Box 1668 Los Alamos, NM 87545
1	Director Sandia National Laboratories Department 913 ATTN: Ron Andreas Albuquerque, NM 87185-5800
1	Director Sandia National Laboratories Division 1611 ATTN: Tom James Albuquerque, NM 87185
1	Director Sandia National Laboratories Division 1623 ATTN: Larry Hostetler Albuquerque, NM 87185
1	Director Sandia National Laboratories ATTN: Gary W. Richter PO Box 969 Livermore, CA 94550
1	Commander Naval Air Systems Command JTCG/AS Central Office ATTN: 5164J (LTC James B. Sebolka) Washington, DC 20361

No. of Copies | Organization

1 Commander
ADR Program Manager
CODE AIR-41112I
ATTN: Tom Furlough
Naval Air Systems Command
Washington, DC 20361-4110

1 Commander
Naval Ocean Systems Center
ATTN: Earle G. Schweizer
Code 000
San Diego, CA 92151-5000

5 Commander
Naval Surface Warfare Center
ATTN: Gregory J. Budd
James Ellis
Barbara J. Harris
Constance P. Rollins
Tom Wasmund
Code G13
Dahlgren, VA 22448-5000

1 Commander
Naval Surface Warfare Center
ATTN: Glen Hornbaker
Code G102
Dahlgren, VA 22448-5000

1 Commander
Naval Surface Warfare Center
ATTN: George Williams
Code J33
Dahlgren, VA 22448-5000

1 Commander
Naval Surface Warfare Center
ATTN: Frank Fassnacht
10901 New Hampshire Ave.
Code N15
Silver Spring, MD 20903-5000

1 Commander
Naval Surface Warfare Center
ATTN: Norma D. Holland
Code R-14
Silver Spring, Md 20903-5000

No. of Copies | Organization

1 Commander
Naval Surface Warfare Center
ATTN: Dr. F.E. Baker
10901 New Hampshire Ave.
Silver Spring, MD 20903-5000

1 Commander
Naval Surface Warfare Center
ATTN: William Emberson
Code H021
10901 New Hampshire Avenue
Silver Spring, MD 20903-5000

1 Commander
Naval Surface Warfare Center
ATTN: M. John Timo
10509 Edgefield Drive
Adelphi, MD 20783-1130

2 Commander
U.S. Naval Weapons Center
ATTN: Jay Butterworth
Dr. Helen Wang
Code 3951
Bldg 1400, Room B20
China Lake, CA 93555-6001

1 Commander
U.S. Naval Weapons Center
ATTN: David H. Hall (Code 3181)
China Lake, CA 93555-6001

1 Commander
U.S. Naval Weapons Center
ATTN: Mark D. Alexander (Code 3894)
China Lake, CA 93556-6001

1 Commander
U.S. Naval Weapons Center
ATTN: Melvin H. Keith (Code 35101)
China Lake, CA 93555-6001

1 Commander
U.S. Naval Weapons Center
ATTN: Tim Horton (Code 3386)
China Lake, CA 93555-6001

No. of Copies	Organization
1	Commander U.S. Naval Weapons Center ATTN: Robert Cox, Code 3517 China Lake, CA 93555-6001
1	Commander U.S. Naval Civil Eng Laboratories ATTN: John M. Ferritto (Code L53) Port Hueneme, CA 93043
1	Naval Postgraduate School ATTN: Professor Robert E. Ball Department of Aeronautics and Astronautics Monterey, CA 93943
1	Naval Postgraduate School ATTN: Dr. Michael J. Zyda Department of Computer Science Code 52 Monterey, CA 93943-5000
1	Naval Postgraduate School Department of National Security ATTN: Dr. Joseph Sternberg Code 73 Monterey, CA 93943
1	Commander Naval Air Systems Command ATTN: Mr. Philip Weinberg Code AIR-516J Washington, DC 20361-5160
1	Commander Naval Sea Systems Command ATTN: William G. Boyce Code 56Y52 Washington, DC 20362
1	Commander Naval Sea Systems Command ATTN: Granville W. Broome SEA 5011 2521 Jefferson Davis Hwy. Arlington, VA 22202

No. of Copies	Organization
1	Commander Naval Sea Systems Command ATTN: Philip M. Covich SEA 55X Washington, DC 20362-5101
2	Commander Naval Sea Systems Command ATTN: CPT Charles Calvano USN Robert Keane, Jr. SEA 50 Washington, DC 20362-5101
1	Commander Naval Sea Systems Command ATTN: Oliver F. Braxton 2521 Jefferson Davis Hwy. Arlington, VA 22202
1	Commander Naval Sea Systems Command ATTN: Donald Ewing Code 503 2521 Jefferson Davis Hwy. Arlington, VA 22202
1	Commander Naval Sea Systems Command ATTN: Larrie D. Ferreiro SEA 501 2521 Jefferson Davis Hwy. Arlington, VA 22202
1	Commander Naval Sea Systems Command ATTN: Anthony F. Johnson SEA 05R2 Washington, DC 20362-5101
1	Commander Naval Sea Systems Command ATTN: CPT William E. Mahew USN PMS 423 Washington, DC 20362-5101

No. of Copies	Organization
1	Commander Naval Sea Systems Command ATTN: Carl H. Pohler Code 05R23 Washington, DC 20362-5101
1	Commander Naval Sea Systems Command ATTN: Ronald P. Kramer SEA 50143 2521 Jefferson Davis Hwy. Arlington, VA 22202
1	Commander Naval Sea Systems Command ATTN: CPT R. Percival USN SEA 05T 2521 Jefferson Davis Hwy. Arlington, VA 22202
1	Commander Space and Naval Warfare Systems Command ATTN: Paul Wessel Code 30T Washington, DC 20363-5100
1	Commander Intelligence Threat Analysis Center ATTN: PSD-GAS/John Bickle Washington Navy Yard Washington, DC 20374
1	Commander Intelligence Threat Analysis Center ATTN: Ron Demeter Washington Navy Yard, B-213, Stop 314 Washington, DC 20374
1	Commander Intelligence Threat Analysis Center ATTN: Tim Finnegan Washington Navy Yard, B-213 Washington, DC 20374

No. of Copies	Organization
2	Commander David W. Taylor Naval Ship and Development Center ATTN: W. Conley J. Schot Bethesda, MD 20084
1	Office of Naval Technology ATTN: David J. Siegel 800 N. Quincy Street Arlington, VA 22217-5000
1	Commander Eglin Air Force Base AD/ENL ATTN: Robert L. Stovall Eglin AFB, FL 32542
1	Commander USAF HQ ESD/PLEA Chief, Engineering and Test Division ATTN: Paul T. Courtoglous Hanscom AFB, MA 01730
2	Commander AFATL ATTN: AGA (Lawrence Jones) (Mickie Phipps) Eglin AFB, FL 32542-5434
1	WL/MNMW (Mr. John A. Collins) Eglin AFB, FL 32542
1	Commander AFEWC ATTN: AFEWC/SAXE (Bod Eddy) Kelly AFB, TX 78243-5000
1	Commander AFWAL/AARA ATTN: Ed Zelano Wright-Patterson AFB, OH 45433
1	Commander AFWAL/FIES ATTN: James Hodges Sr. Wright-Patterson AFB, OH 45433-6523

No. of Copies	Organization
2	Commander AFWAL/MLTC ATTN: LT Robert Carringer Dave Judson Wright-Patterson AFB, OH 45433-6533
1	Commander ASB/XRM ATTN: Gerald Bennett Martin Lentz Wright-Patterson AFB, OH 45433
1	Commander WRDC/AARA ATTN: Michael L. Bryant Wright-Patterson AFB, OH 45433
1	Commander FTD/SDMBA ATTN: Charles Darnell Wright-Patterson AFB, OH 45433
1	Commander FTD/SDMBU ATTN: Kevin Nelson Wright-Patterson AFB, OH 45433
1	Commander FTD/SQDRA ATTN: Greg Koesters Wright-Patterson AFB, OH 45433-6508
1	Commander FTD ATTN: Tom Reinhardt Wright-Patterson AFB, OH 45433
1	Commander FTD/SDAEA ATTN: Joe Sugrue Wright-Patterson AFB, OH 45433
1	Commander AFWAL/AARA ATTN: Vincent Velten Wright-Patterson AFB, OH 45433

No. of Copies	Organization
1	Commander FTD/SQDRA ATTN: Larry E. Wright Wright-Patterson AFB, OH 45433
1	Commander AD/CZL ATTN: James M. Heard Eglin AFB, FL 32542-5000
1	Commander AD/ENY ATTN: Dr. Stewart W. Turner Director of Engineering Analysis Eglin AFB, FL 32542-5000
2	Commander AD/ENYW ATTN: 2LT Michael Ferguson Jim Richardson Eglin AFB, FL 32542-5000
1	Commander Air Force Armament Laboratory ATTN: AFATL/DLY (James B. Flint) Eglin AFB, FL 32542-5000
1	Commander U.S. Army FSTC/CA3 ATTN: Scott Mingledorff 220 Seventh Avenue Charlottesville, VA 22901-5396
1	Commander U.S. Army FSTC (UK) ATTN: MAJ Nigel Williams 220 Seventh Avenue Charlottesville, VA 22901-5396
1	Commander U.S. Army FSTC ATTN: Dr. Tim Small 220 Seventh Avenue Charlottsville, VA 22901-5396
1	Defense Intelligence Agency ATTN: DB-6E3 (Jay Hagler) Washington, DC 20340-6763

No. of Copies	Organization
5	Institute for Defense Analyses (IDA) ATTN: Mr. Irwin A. Kaufman Mr. Arthur O. Kresse Dr. Lowell Tonnessen Mr. Benjamin W. Turner Ms. Sylvia L. Waller 1801 N. Beauregard Street Alexandria, VA 22311
1	Institute for Defense Analyses ATTN: Carl F. Kossack 1005 Athens Way Sun City, FL 33570
1	Institute for Defense Analyses ATTN: Dr. Natarajan Subramonian 14309 Hollyhock Way Burtonsville, MD 20866
1	Department of Commerce National Institute of Standards and Technology Manufacturing Systems Group ATTN: B. Smith Washington, DC 20234
1	AAI Corporation ATTN: H. W. Schuette PO Box 126 Hunt Valley, MD 21030-0126
2	ADPA ATTN: Donna R. Alexander Bill King Two Colonial Place, Suite 400 2101 Wilson Boulevard Arlington, VA 22201-3061
1	ARC Professional Services Group ATTN: Arnold R. Gritzke 5501 Backlick Road Springfield, VA 22151
2	Advanced Marine Enterprises ATTN: James F. Hess CPT Frederic S. Hering USN (Ret) 1725 Jefferson Davis Highway Suite 1300 Arlington, VA 22202

No. of Copies	Organization
1	The Armed Forces Communications and Electronics Association ATTN: Kirby Lamar, BG(Ret) 4400 Fair Lakes Court Fairfax, VA 22033-3899
2	Aero Corporation ATTN: David S. Eccles Gregg Snyder P.O. Box 92957, M4/913 Los Angeles, CA 90009
1	AFELM, The Rand Corporation ATTN: Library-D 1700 Main Street Santa Monica, CA 90406
2	Air Force Wright Aeronautical Labs ATTN: CDJ, CPT Jost CDJ, Joseph Faison Wright-Patterson AFB, OH 45433-6523
1	Alliant Techsystems, Inc. ATTN: Hatem Nasr Systems and Research Center 3660 Technology Drive P.O. Box 1361 Minneapolis, MN 55418
1	Alliant Techsystems, Inc. ATTN: Fred J. Parduhn 7225 Northland Drive Brooklyn Park, MN 55428
2	Alliant Techsystems, Inc. ATTN: Raymond H. Burg Laura C. Dillway MN38-4000 10400 Yellow Circle Drive Minnetonka, MN 55343
1	Allison Gas Turbine Division of GM ATTN: Michael Swift PO Box 420, SC S22B Indianapolis, IN 46260-0420

No. of Copies | Organization

1 Aluminum Company of America
ATTN: Frank W. Baker
Alcoa Technical Center
Alcoa Center, PA 15069

1 Analysis and Technology
ATTN: RADM Thomas M. Hopkins USN (Ret)
1113 Carper Street
McLean, VA 22101

1 ANSER
ATTN: James W. McNulty
1215 Jefferson Davis Highway
Arlington, VA 22202

1 ARC C-500
ATTN: John H. Bucher
Modena Road
Coatesville, PA 19320

1 Armament Systems, Inc.
ATTN: Gerard Zeller
P.O. Box 158
211 West Bel Air Avenue
Aberdeen, MD 21001

1 Armored Vehicle Technologies
ATTN: Coda M. Edwards
PO Box 2057
Warren, MI 48090

1 Army High Performance Computing Research Center / Systems
ATTN: Dennis R Lienke
Minnesota Supercomputer Center
1200 Washington Avenue South
Minneapolis, Minnesota 55415

1 ASI Sytems, International
ATTN: Dr. Michael Stamatelatos
3319 Lone Jack Road
Encinitas, CA 92024

1 Auburn University
Electrical Engineering Department
ATTN: Dr. Thomas Shumpert
Auburn University, AL 36849

No. of Copies | Organization

1 A.W. Bayer and Associates
ATTN: Albert W. Bayer, President
Marina City Club
4333 Admiralty Way
Marina del Rey, CA 90292-5469

1 Battelle
ATTN: TACTEC Library (J.N. Huggins)
505 King Avenue
Columbus, OH 43201-2693

1 Battelle
Defense and Space Systems Analysis
ATTN: Dr. Richard K. Thatcher
505 King Avenue
Columbus, OH 43201-2693

1 Battelle
ATTN: Bernard J. Tullington
1300 N. 17th Street, Suite 1520
Arlington, VA 22209

3 Battelle
Edgewood Operations
ATTN: Roy Golly
Gene Roecker
Robert Jameson
2113 Emmorton Park Road
Edgewood, MD 21040

1 The BDM Corporation
ATTN: Edwin J. Dorchak
7915 Jones Branch Drive
McLean, VA 22102-3396

1 The BDM Corporation
ATTN: Fred J. Michel
1300 N. 17th Street
Arlington, VA 22209

1 Bell Helicopter, Textron
ATTN: Jack R. Johnson
PO Box 482
Fort Worth, TX 76101

No. of Copies | Organization

3 BMY, Division of Harsco
ATTN: William J. Wagner, Jr.
Ronald W. Jenkins
Ed Magalski
PO Box 1512
York, PA 17404

1 Board on Army Science and Technology
National Research Council
Room MH 280
2101 Constitution Avenue, NW
Washington, DC 20418

2 Boeing Aerospace
ATTN: Dr. Robert Chiavetta
Dr. John Kuras
Mail Stop 8K17
P.O. Box 3999
Seattle, WA 98124-2499

1 Boeing Military Airplanes
ATTN: MS K80-08, Jerry White
PO Box 7730
Wichita, KA 67277-7730

1 Boeing Vertol Company
A Division of Boeing Co.
ATTN: MS P30-27, John E. Lyons
PO Box 16858
Philadelphia, PA 19142

1 Booz-Allen and Hamilton, Inc.
ATTN: Dr. Richard B. Benjamin
Suite 131, 4141 Colonel Glenn Hwy.
Dayton, OH 45431

1 Booz-Allen and Hamilton, Inc.
ATTN: Lee F. Mallett
1300 N. 17th Street, Suite 1610
Rosslyn, VA 22209

2 Booz-Allen and Hamilton, Inc.
ATTN: John M. Vice
WRDC/FIVS/SURVIAC
Bldg 45, Area B
Wright-Patterson AFB, OH 45433-6553

No. of Copies | Organization

1 John Brown Associates
ATTN: Dr. John A. Brown
PO Box 145
Berkeley Heights, NJ 07922-0145

1 Chamberlain
ATTN: Mark A. Sackett
PO Box 2545
Waterloo, IA 50704

1 Commander
Combined Arms Combat Development
ATTN: ATZL-CAP (LTC Morrison
Dir, Surv Task Force)
Ft. Leavenworth, KS 66027-5300

1 Commander
Combined Arms Combat Development
ATTN: ATZL-HFM (Dwain Skelton)
Ft. Leavenworth, KS 66027-5300

1 Computer Sciences Corporation
ATTN: Abner W. Lee
200 Sparkman Drive
Huntsville, AL 35805

1 CRS Sirrine, Inc.
ATTN: Dr. James C. Smith
PO Box 22427
1177 West Loop South
Houston, TX 77227

2 Cypress International
ATTN: August J. Caponecchi
James Logan
1201 E. Abingdon Drive
Alexandria, VA 22314

1 DATA Networks, Inc.
ATTN: William E. Regan, Jr.
President.
288 Greenspring Station
Brooklandville, MD 21022

1 DNA
ATTN: LCDR Charles Nofziger
6801 Telegraph Road
Alexandria, VA 22310

No. of Copies	Organization
1	Datatec, Inc. ATTN: Donald E. Cudney President 326 Green Acres Fort Walton, FL 32548
1	David Taylor Research Center ATTN: Dr. Fred J. Fisch 2203 Eastlake Road Timonium, MD 21093-5000
1	David Taylor Research Center ATTN: Robert E. Fuss UERD, Code 177 Portsmouth, VA 23709-5000
1	David Taylor Research Center ATTN: Seymour N. Goldstein Code 1210 Bethesda, MD 20084-5000
1	David Taylor Research Center ATTN: Ib S. Hansen Code 174 Bethesda, MD 20084-5000
1	David Taylor Research Center ATTN: Harry Price Gray Code 1740.4 Bethesda, MD 20084-5000
1	David Taylor Research Center ATTN: Jackson T. Hawkins Code 1740.2 Bethesda, MD 20084-5000
1	David Taylor Research Center ATTN: Steven L. Cohen Code 1230 Bethesda, MD 20084-5000
1	David Taylor Research Center ATTN: Dennis Clark Code 0111 Bethesda, MD 20084-5000

No. of Copies	Organization
1	David Taylor Research Center ATTN: John R. Krezel UERD, Code 177.2 Portsmouth, VA 23709-5000
1	David Taylor Research Center ATTN: Richard E. Metrey Code 01 Bethesda, MD 20084-5000
1	David Taylor Research Center ATTN: Dr. Paul C. St. Hilaire Code 1210 Bethesda, MD 20084-5000
1	David Taylor Research Center ATTN: Arthur Marchand Code 2843 Annapolis, MD 21042
1	David Taylor Research Center ATTN: Michael Riley UERD, Code 177 Portsmouth, VA 23709-5000
1	David Taylor Research Center ATTN: J. William Sykes Code 175 Bethesda, MD 20084-5000
1	David Taylor Research Center ATTN: Herbert Wolk Code 1740.1 Bethesda, MD 20084-5000
1	University of Dayton Graduate Engineering and Research Kettering Lab 262 ATTN: Dr. Gary Thiele, Director Dayton, OH 45469
1	Defense Nuclear Agency Structural Dynamics Section ATTN: Tom Tsai Washington, DC 20305

No. of Copies | Organization

1 Delco Systems Operation
ATTN: John Steen
6767 Hollister Avenue, #P202
Goleta, CA 93117

1 Denver Research Institute
BW 228
ATTN: Lawrence G. Ullyatt
2050 E. Iliff Avenue
Denver, CO 80208

1 Dow Chemical, U.S.A
ATTN: Dr. P. Richard Stoesser
Contract R&D
1801 Building
Midland, MI 48674-1801

1 Drexel University
ATTN: Dr. Pei Chi Chou
College of Engineering
Philadelphia, PA 19104

1 DuPont Company FPD
ATTN: Dr. Oswald R. Bergmann
B-1246, 1007 Market Street
Wilmington, DE 19898

1 Dynamics Analysis and Test Associates
ATTN: Dr. C. Thomas Savell
2231 Faraday Ave
Suite 103
Carlsbad, CA 92008

1 E. I. Dupont TED FMC
ATTN: Richard O. Myers Jr.
Wilmington, DE 19898

1 Eichelberger Consulting Company
ATTN: Dr. Robert Eichelberger
President
409 West Catherine Street
Bel Air, MD 21014

1 Electronic Warfare Associates, Inc.
ATTN: William V. Chiaramonte
2071 Chain Bridge Road
Vienna, VA 22180

No. of Copies | Organization

1 Emprise, Ltd.
ATTN: Bradshaw Armendt, Jr
201 Crafton Road
Bel Air, MD 21014

8 Environmental Research Institute of Michigan
ATTN: Mr. K. Augustyn
Mr. Kozma
Dr. I. La Haie
Mr. R. Horvath
Mr. Arnold
Mr. E. Cobb
Mr. B. Morey
Mr. M. Bair
PO Box 8618
Ann Arbor, MI 48107

1 E-OIR Measurements, Inc.
ATTN: Russ Moulton
PO Box 1240
Spotsylvania, VA 22553-1240

1 ERIM
ATTN: Stephen R. Stewart
Exploitation Applications Department
Image Processing Systems Division
PO Box 8618
Ann Arbor, MI 48107-8618

1 USA ETL/IAG
ATTN: Jim Campbell
Bldg 2592, Room S18
Ft. Belvoir, VA 22060-5546

1 FMC Corporation
ATTN: Sidney Kraus
1105 Coleman Ave, Box 1201
San Jose, CA 95108

3 FMC Corporation
ATTN: Ronald S. Beck
Martin Lim
Jacob F. Yacoub
881 Martin Avenue
Santa Clara, CA 95052

No. of Copies — Organization

5 FMC Corporation
Advanced Systems Center (ASC)
ATTN: Charles A. Millard
Scott L. Langlie
Herb Theumer
Walter L. Davidson
J.E. Alexander
1300 South Second Street
PO Box 59043
Minneapolis, MN 55459

1 BDM International
ATTN: Mr. Steve Church, FX2B307
7915 Jones Branch Drive
McLean, VA 22102-3396

1 BDM International
ATTN: Mr. Tom Hooker, FF2B304
7915 Jones Branch Drive
McLean, VA 22102-3396

2 FMC Corporation
Defense Systems Group
ATTN: Robert Burt
Dennis R. Nitschke
1115 Coleman Avenue
San Jose, CA 95037

1 FMC Corporation
Naval Systems Division (NSD)
ATTN: MK-45, Randall Ellis
Minneapolis, MN 55421

1 FMC Corporation
Northern Ordnance Division
ATTN: M3-11, Barry Brown
4800 East River Road
Minneapolis, MN 55421

6 FMC Corporation
Ordnance Engineering Division
ATTN: H. Croft
M. Hatcher
L. House
J. Jackson
E. Maddox
R. Musante
1105 Coleman Ave, Box 1201
San Jose, CA 95108

No. of Copies — Organization

1 GE Aircraft Engines
ATTN: Dr. Roger B. Dunn
One Neumann Way, MD J185
Cincinnati, OH 45215-6301

1 General Atomics
ATTN: Chester J. Everline,
Staff Engineer
P.O. Box 85608
San Diego, CA 92138-5608

1 General Dynamics
ATTN: Dr. Fred Cleveland
P.O. Box 748
Mail Zone 5965
Ft. Worth, TX 76101

3 General Dynamics
ATTN: MZ-4362112, Robert Carter
MZ-4362029, Jim Graciano
MZ-4362055, Gary Jackman
38500 Mound
Sterling Heights, MI 48310

3 General Dynamics Corporation
ATTN: MZ-2650, Dave Bergman
MZ-2860, John Romanko
MZ-2844, Cynthia Waters
PO Box 748
Ft. Worth, TX 76101-0748

1 General Dynamics Land Systems
ATTN: Robert Carter
PO Box 1804
Warren, MI 48090-2074

1 General Dynamics Land Systems
ATTN: Dr. Paulus Kersten
PO Box 1901
Warren, MI 48090-2074

1 General Dynamics Land Systems
ATTN: William M. Mrdeza
PO Box 2045
Warren, MI 48090-2074

No. of Copies | Organization

1 General Dynamics Land Systems
ATTN: Jay A. Lobb
PO Box 2074, Mail Zone 436-21-19
Warren, MI 48090-2074

5 General Dynamics Land Systems
ATTN: Richard Auyer
Otto Renius
N. S. Sridharan
Dean R. Loftin
Dr. Phil Lett
PO Box 2074
Warren, MI 48090-2074

3 General Motors Corporation
Research Laboratories
ATTN: J. Boyse
J. Joyce
R. Sarraga
Warren, MI 48090

1 Allison Gas Turbine Division
General Motors Corporation
ATTN: John A. MacBain, Ph.D., Supervisor
Low Observables Technology
Propulsion Systems Integration
PO Box 420, Speed Code W-16
Indianapolis, IN 46206-0420

1 Geometric Solutions, Inc.
ATTN: Harry Reed
100 Custis Street
Suite 3
P. O. Box 382
Aberdeen, MD 21001

1 Gettysburg College
Box 405
Gettysburg, PA 17325

1 GTRI-RAIL-MAD
ATTN: Mr. Joe Bradley
CRB 577
Atlanta, GA 30332

No. of Copies | Organization

1 Hughes Associates
ATTN: J. Thomas Hughes
2730 University Blvd.
Suite 902
Wheaton, MD 20902

2 INEL/EG&G
Engineer Lab
ATTN: Ray Berry
M. Marx Hintze
PO Box 1625
Idaho Falls, ID 83451

1 Interactive Computer Graphics Center
Rensselear Polytechnic Inst.
ATTN: M. Wozny
Troy, NY 12181

1 International Development Corporation
ATTN: Trevor O. Jones, President
One Cleveland Center, Suite 2900
1375 East Ninth Street
Cleveland, OH 44114-1724

1 Intergraph
National Exploitation Systems
ATTN: John H. Suter
2051 Mercator Drive
Reston, VA 22091-3413

1 ISAT
ATTN: Roderick Briggs
1305 Duke Street
Alexandria, VA 22314

1 ITT Defense
ATTN: Joseph Conway
1000 Wilson Blvd.
30th Floor
Arlington, VA 22209

1 Joint Technical Coordinating Group
ATTN: Philip Weinberg
JTCG/AS5
AIR-516J5
Washington, DC 20361-5160

No. of Copies	Organization
1	California Institute of Technology Jet Propulsion Laboratory ATTN: D. Lewis 4800 Oak Grove Drive Pasadena, CA 91109
1	Kaman Sciences Corporation ATTN: Timothy S. Pendergrass 600 Boulevard South, Suite 208 Huntsville, AL 35802
1	Ketron, Inc. ATTN: Robert S. Bennett 901 Dulaney Valley Rd, Suite 220 Baltimore, MD 21204-2600
1	Keweenaw Research Center Michigan Technological University ATTN: Bill Reynolds Houghton, MI 49931
1	Lanxido Armor Products ATTN: Dr. Robert A. Wolffe Tralee Industrial Park Newark, DE 19711
2	Lincoln Laboratory MIT ATTN: Dr. Robert Shin Dr. Chuck Burt P.O. Box 73 Lexington, MA 02173
3	Lincoln Laboratory MIT Surveillance Systems Group ATTN: R. Barnes G. Knittel J. Kong 244 Wood Street Lexington, MA 02173-0073
3	Lockheed-California Company ATTN: C. A. Burton R. J. Ricci M. Steinberg Burbank, CA 91520
2	Lockheed-Georgia Company ATTN: Ottis F. Teuton J. Tulkoff Dept. 72-91, Zone 419 Marietta, GA 30063
1	Lockheed Palo Alto Research Lab ATTN: John A. DeRuntz, JR 0/93, B/25I 3251 Hanover Street Palo Alto, CA 94304
1	Logistics Management Institute ATTN: Edward D. Simms Jr. 6400 Goldsboro Road Bethesda, MD 20817-5886
1	Los Alamos Technical Associates, Inc. ATTN: John S. Daly 6501 Americas Parkway, #900 Albuquerque, NM 87110
2	Los Alamos Technical Associates, Inc. ATTN: James C. Jacobs Donald M. Lund 8550 Arlington Boulevard Suite 301 Fairfax, VA 22031
1	Los Alamos Technical Associates, Inc. ATTN: Thomas Giacofci 3020 Hamaker Court Fairfax, VA 22031
1	LTV Aerospace and Defense Company ATTN: Daniel M. Reedy PO Box 655907 Dallas, TX 75265-5907
3	Martin Marietta Aerospace ATTN: MP-113, Dan Dorfman MP-433, Richard S. Dowd MP-243, Thomas C. D'Isepo PO Box 555837 Orlando, FL 32855-5837

No. of Copies	Organization
1	Maxwell Laboratories, Inc. ATTN: Dr. Michael Holland 8888 Balboa Avenue San Diego, CA 92123-1506
1	McDonnell Douglas Astronautic ATTN: Nikolai A. Louie 5301 Bolsa Avenue Huntington Beach, CA 92647
1	McDonnell Douglas, Inc. ATTN: David Hamilton PO Box 516 St. Louis, MO 63166
1	McDonnell Douglas, Inc. ATTN: Alan R. Parker 3855 Lakewood Blvd., MC 35-18 Long Beach, CA 90846
1	Micro Electronics of North Carolina ATTN: Gershon Kedem PO Box 12889 Research Triangle Park, NC 07709
1	MIT ATTN: Dr. S. Benton RE15-416 Cambridge, MA 02139
6	The MITRE Corporation ATTN: Mr. Edward C. Brady, Vice President Dr. Robert Henderson Dr. Nicklas Gramenopoulos Dr. Narayana Srinivasan Mr. Norman W. Huddy Dr. John M. Ruddy 7525 Colshire Drive McLean, VA 22102-3184
2	NFK Engineering, Inc. ATTN: Dr. Michael P. Pakstys Justin W. Held 4200 Wilson Blvd. Arlington, VA 22203-1800

No. of Copies	Organization
1	NFK Engineering, Inc. ATTN: John J. Turner 1125 Trotting Horse Lane Great Falls, VA 22066
1	NASA-Ames Research Center ATTN: Dr. Alex Woo Mail Stop 227-2 Moffett Field, CA 94035-1000
1	NASA-Ames Research Center ATTN: Leroy Presley Mail Stop 227-4 Moffett Field, CA 94035-1000
1	NAVIR DEVCON ATTN: Frank Wenograd Code 6043 Walminstor, PA 18974
1	North Aircraft ATTN: Dr. Athanosis Varvatsis Mail Zone 3622/84 1 Northrop Ave Hawthorne, CA 90250
1	Northrop Research and Technology Center ATTN: Dr. David Donovan Garber One Research Park Palos Verdes Peninsula, CA 90274
1	Norton Company ATTN: Ronald K. Bart 1 New Bond Street Worcester, MA 01606-2698
1	The Oceanus Company ATTN: RADM Robert H. Gormley, (Ret) PO Box 7069 Menlo Park, CA 94026
1	Oklahoma State University College of Engineering, Architecture and Technology ATTN: Thomas M. Browder, Jr. PO Box 1925 Eglin AFB, FL 32542

No. of Copies	Organization
1	Pacific Scientific/Htl Division ATTN: Robert F. Aldrich 1800 Highland Avenue Duarte, CA 91010
1	Perceptronics, Inc. ATTN: Dean R. Loftin 21111 Erwin Street Woodland Hills, CA 91367
1	Physics Mathematics and Computers, Inc. ATTN: Matt Perini PO Box 787 Socorro, NM 87801
1	Princeton University Mathematics Department Fine Hall Washington Road ATTN: John Tukey Princeton, NJ 08544-1000
1	PRI, Inc. ATTN: W. Bushell Building E4435, Second Floor Edgewood Area-APG, MD 21010
1	RGB Associates, Inc. ATTN: R. Barakat Box B Wayland, MA 01778
1	Rockwell International Corporation ATTN: Dr. H. Bran Tran P.O. Box 92098 Department 113/GB01 Los Angeles, CA 90009
1	Rockwell International Corporation ATTN: Keith R. Rathjen, Vice President 3370 Miraloma Avenue (031-HA01) Anaheim, CA 92803-3105
1	Rome Air Development Center ATTN: RADC/IRRE, Peter J. Costianes Griffis Air Force Base, NY 13441-5700

No. of Copies	Organization
1	Rome Air Development Center RADC/OCTM ATTN: Edward Starczewski Building 106 Griffis Air Force Base, NY 13441-5700
1	S-Cubed ATTN: Michael S. Lancaster 1800 Diagonal Road, Suite 420 Alexandria, VA 22314
1	Sachs/Freeman Associates, Inc. ATTN: Donald W. Lynch Senior Research Physicist 205 Yoakum Parkway, #511 Alexandria, VA 22304
1	SAIC ATTN: Dr. Alan J. Toepfer 2109 Air Park Drive, SE Albuquerque, NM 87106
1	SAIC ATTN: John H. McNeilly, Senior Scientist 1710 Goodridge Drive McLean, VA 22102
2	SAIC ATTN: Terry Keller Robert Turner Suite 200 1010 Woodman Drive Dayton, OH 45432
1	SAIC ATTN: David R. Garfinkle Malibu Canyon Business Park 26679 W. Agoura Road, Suite 200 Calabasas, CA 91302
2	George Sharp Company ATTN: Dennis M. McCarley Roger O. Mau 2121 Crystal Drive Suite 714 Arlington, VA 22202

No. of Copies | Organization

1 Sidwell-Ross and Associates, Inc.
ATTN: LTG Marion C. Ross,
(USA Ret)
Executive Vice President
PO Box 88531
Atlanta, GA 30338

1 Sigma Research Inc.
ATTN: Dr. Richard Bossi
4014 Hampton Way
Kent, WA 98032

1 Simula, Inc.
ATTN: Joseph W. Coltman
10016 South 51st Street
Pheonix, AZ 85044

1 SimTech
ATTN: Dr. Annie V. Saylor
3307 Bob Wallace Ave., Suite 4
Huntsville, AL 35807

1 Alan Smolen and Associates, Inc.
ATTN: Alan Smolen, President
One Cynthia Court
Palm Coast, FL 32027-8172

3 Southwest Research Institute
ATTN: Martin Goland
Alex B. Wenzel
Patrick H. Zabel
P.O. Drawer 28255
San Antonio, TX 78238-0255

3 Sparta, Inc.
ATTN: David M. McKinley
Robert E. O'Connor
Karen M. Rooney
4901 Corporate Drive
Huntsville, AL 35805-6201

1 SRI International
ATTN: Donald R. Curran
333 Ravenswood Ave.
Menlo Park, CA 94025

No. of Copies | Organization

1 Star Laboratory, Stanford University
ATTN: Dr. Joseph W. Goodman
Electrical Engineering Department
233 Durand Building
Stanford, CA 94305-4055

1 University of Michigan
ATTN: Dr. John F. Vesecky
2212 Space Research Blvd.
Ann Arbor, MI 48109-2143

1 Princeton University
ATTN: Dr. Curt Callen
Physics Department
PO Box 708
Princeton, NJ 08544

1 University of California, San Diego
ATTN: Dr. Gordon J. MacDonald
Institute on Global Conflict
and Cooperation (0518)
9500 Gilman Drive
La Jolla, CA 92093-0518

3 Structural Dynamics Research
Corporation (SDRC)
ATTN: R. Ard
W. McClelland
J. Osborn
2000 Eastman Drive
Milford, OH 45150

1 Syracuse Research Group
ATTN: Dr. Chung-Chi Cha
Merrill Lane
Syracuse, NY 13210

1 System Planning Corporation
ATTN: Ann Hafer
1500 Wilson Blvd
Arlington, VA 22209

1 S-Cubed
ATTN: Robert T. Sedgwick
PO Box 1620
La Jolla, CA 92038-1620

No. of Copies | Organization

2 TASC
ATTN: Richard E. Kinsler
Darrell James
970 Mar-Walt Drive
Ft. Walton Beach, FL 32548

1 TASC
ATTN: Harry I. Nimon, Jr
1700 N. Moore Street, Suite 1220
Arlington, VA 22209

1 TASC
ATTN: COL James Logan (Ret)
1101 Wilson Blvd.
Suite 1500
Arlington, VA 22209

1 COLSA, Inc.
ATTN: Mr. Willy Albanes
P.O. Box 1068
Huntsville, AL 35807-3301

1 Techmatics, Inc.
ATTN: Ronald R. Rickwald
2231 Crystal Drive
Arlington, VA 22202-3742

1 Technical Solutions, Inc
ATTN: John R. Robbins
P.O. Box 1148
Mesillia Park, NM 88047

1 Teledyne Brown Engineering
ATTN: John W. Wolfsberger, Jr.
Cummings Research Park
300 Sparkman Drive, NW
PO Box 070007
Huntsville, AL 35807-7007

4 The SURVICE Engineering Co.
ATTN: Jim Foulk
George Large
Glenn Gillis
Kris Keller
Suite 103
1003 Old Philadelphia Road
Aberdeen, MD 21001

No. of Copies | Organization

1 Tradeways, Ltd.
ATTN: Joseph G. Gorski,
President
307F Maple Avenue West
Vienna, VA 22180

1 Ultramet
ATTN: Dr. Jacob J. Stiglich
12173 Montague Street
Pacoima, CA 91331

1 United Technologies Corporation
Advanced Systems Division
ATTN: Richard J. Holman
10180 Telesis Court
San Diego, CA 92121

1 University of Idaho
Department of Civil Engineering
ATTN: Dr. Dennis R. Horn
Assistant Professor
Moscow, ID 83843-4194

1 University of Illinois at Chicago
Communications Laboratory
ATTN: Dr. Wolfgang-M. Boerner
PO Box 4348
M/C 154, 1141-SEO
Chicago, IL 60680

1 University of Illinois at Urbana-Champaign
Department of Civil Engineering
and Environmental Studies
ATTN: Dr. E. Downey Brill, Jr.
208 North Romine
Urbana, IL 61801-2374

1 University of Illinois at Urbana-Champaign
Department of Electrical and Computer
Engineering
ATTN: Dr. Shung-Wu Lee
1406 W. Green
Urbana, IL 61801

1 The Johns Hopkins University
Applied Physics Laboratory
ATTN: Jonathan Fluss
Johns Hopkins Road
Laurel, MD 20707

No. of Copies	Organization
1	University of Nevada Environmental Research Center ATTN: Dr. Delbert S. Barth Senior Scientist Las Vegas, NV 89154-0001
1	University of North Carolina ATTN: Professor Henry Fuchs 208 New West Hall (035A) Chapel Hill, NC 27514
3	Ohio State University Electroscience Laboratory ATTN: Dr. Ronald Marhefka Dr. Edward H. Newman Dr. Prasbhaker H. Pathak 1320 Kinnear Road Columbus, OH 43212
1	University of Rochester ATTN: Nicholas George College of Engineering and Applied Science Rochester, NY 14627
3	University of Utah Computer Science Department ATTN: R. Riesenfeld E. Cohen L. Knapp 3160 Merrill Engineering Bldg Salt Lake City, UT 84112
3	University of Washington 409 Department of Electrical Engineering, FT-10 ATTN: Dr. Irene Peden Dr. Akira Ishimaru Dr. Chi Ho Chan Seattle, WA 98105
1	Virginia Polytechnic Institute and State University Industrial Engineering Operations Research Department ATTN: Robert C. Williges 302 Whittemore Hall Blacksburg, VA 24061-8603

No. of Copies	Organization
1	LTV Aircraft Products Group ATTN: Paul T. Chan, M/S 194-63 PO Box 655907 Dallas, TX 75265-5907
1	LTV Aerospace and Defense Company LTV Missiles and Electronics Group ATTN: Roger W. Melin PO Box 650003 M/S EM-36 Dallas, TX 75265-0003
1	Wackenhut Applied Technologies Center ATTN: Robert D. Carpenter 10530 Rosehaven St. Suite 500 Fairfax, VA 22030-2877
1	Westinghouse ATTN: Harvey Kloehn Box 1693 MS 8530 Baltimore, MD 21203
1	XMCO, Inc. 460 Spring Park Pl #1500 Herndon, VA 22070-5215
1	Zernow Tech Services, Inc. ATTN: Dr. Louis Zernow 425 West Bonita, Suite 208 San Dimas, CA 91773
2	Sverdrup Technology ATTN: Dr. Ralph Calhoun Bud Bruenning PO Box 1935 Eglin AFB, FL 32542
1	Georgia Technical Research Institute Systems and Technical Laboratory ATTN: Dr. Charles Watt 1770 Richardsons Road Smyrna, GA 30080

No. of Copies	Organization
1	Georgia Institute of Technology ATTN: Dr. Richard Moore ECSL/EME ERB Building, Room 111 Atlanta, GA 30332
1	Georgia Institute of Technology ATTN: Dr. L. G. Callahan, Jr. School of Industrial & Systems Engineering 765 Ferst Drive Atlanta, GA 30332-0385
1	Duke University Department of Computer Science, VLSI Raycasting ATTN: Dr. Gershon Kedem 236 North Building Durham, NC 27706
1	UNISYS Corporation ATTN: Calvin M. Shintani 12010 Sunrise Valley Drive Department 7412 Reston, VA 22091
1	Weidlinger Assoicates, Inc. ATTN: Kenneth Stultz 1735 Jefferson Davis Hwy. Suite 1002 Arlington, VA 22202
1	Mr. Charles W. Bernard 5300 Columbia Pike Apt. #902 Arlington, VA 22204
1	Mr. Michael W. Bernhardt, DA Consultant Rt. 1, 12 Arthur Drive Hockessin, DE 19707
1	Mr. H. G. Bowen Jr., DA Consultant 408 Crown View Drive Alexandria, VA 22314-4804

No. of Copies	Organization
1	Mr. Harvey E. Cale, DA Consultant 2561 Meadowbrook Lane Carson City, NV 89701-5726
1	Perkins Coie ATTN: Mr. Robert L. Deitz 607 Fourteenth Street, NW Washington, DC 20005-2011
1	Dr. F. Paul Carlson DA Consultant 11668 Tanglewood Drive Eden Prairie, MN 55347
1	Mr. Gerard W. Elverum, Jr. 1338 Fair Oaks Avenue Banning, CA 92220
1	Mr. Richard E. Entlich National Research Council 2101 Constitution Ave., NW Harris Building #254 Washington, DC 20418
1	Dr. Edward J. Haug University of Iowa College of Engineering Center for Computer Aided Design Iowa City, IA 52242
1	Mr. Robert M. Hillyer Orincon Corporation 9363 Towne Centre Drive San Diego, CA 92121
1	Dr. Robert B. LaBerge 910 Via Palo Aptos, CA 95003
1	Orr Associates, Inc. ATTN: Dr. Joel N. Orr 5224 Indian River Road Virginia Beach, VA 23464
1	Mr. Abraham Golub DA Consultant 203 Yoakum Parkway, Apt 607 Alexandria, VA 22304

No. of Copies	Organization
1	Mr. Dave Hardison ASB Consultant 3807 Bent Branch Road Falls Church, VA 22041
1	Mr. William M. Hubbard, ASB Consultant 613 Eastlake Drive Columbia, MO 65203
1	Mr. Charles E. Joachim, DA Consultant PO Box 631 Vicksburg, MS 39180
1	Dr. Edward R. Jones, DA Consultant 9881 Wild Deer Road St. Louis, MO 63124
1	MG Robert Kirwan (USA Ret), DA Consultant 10213 Grovewood Way Fairfax, VA 22032
1	Director TEXCOM FSTP ATTN: STE-TFS-Z (Donald J. Krejcarek) Ft. Sill, OK 73503-6100
1	Mr. Robert B. Kurtz, DA Consultant 542 Merwins Lane Fairfield, CT 06430-1920
1	LTGEN Howard W. Leaf USAF (Retired) 8504 Brook Road McLean, VA 22101
1	Dr. Roy A. Lucht, Group M-B MS-J960 Los Alamos, NM 87545
1	Mr. Donald F. Menne, DA Consultant 617 Foxcroft Drive Bel Air, MD 21014

No. of Copies	Organization
1	Mr. Richard C. Messinger Vice President and Chief Technical Officer Cincinnati Mailacron Inc. Cincinnati, OH 45209
1	Mrs. Hyla Napadensky 650 Judson Avenue Evanston, IL 60202-2551
1	Larry Nutsch Sr. Research Engineer 2050 East Iliff Avenue Denver, Colorado 80208
1	GEN Glenn Otis USA (Ret) Coleman Research Corporation 5950 Lakehurst Drive Orlando, FL 32819
1	MG Peter G. Olenchuk (USA Ret), BAST Consultant 6801 Baron Road McLean, VA 22101
1	Mr. Albert E. Papazoni, DA Consultant 1600 Surrey Hill Drive Austin, TX 78746-7338
1	Harry Reed, Sr. Battelle Consultant 138 Edmund Street Aberdeen, MD 21001
1	Mr. David L. Rigotti McClean Research Consultant 127 Duncannon Road Bel Air, MD 21014
1	Dr. A. E. Schmidlin, DA Consultant 28 Highview Road Caldwell, NJ 07006-5502
1	Mr. Robert G.S. Sewell 1236 Mt. Whitney Lane Ridgecrest, CA 93555

No. of Copies | Organization

1 MEMEX Corporation
ATTN: Mr. Charles S. Smith
9 Doaks Lane
Marblehead, MA 01945

1 Mr. Arthur Stein,
Consultant
Institute for Defense Analyses
30 Chapel Woods Court
Williamsville, NY 14221-1816

1 Dr. Dora Strother,
ASB Consultant
3616 Landy Lane
Ft. Worth, TX 76118

1 Mr. Charles F. Tiffany
9 Westerly Lane
Centerville, OH 45458

Aberdeen Proving Ground

18 Dir, USAMSAA
ATTN: AMXSY-A, W. Clifford
J. Meredith
AMXSY-C, A. Reid
W. Braerman
AMXSY-CR, M. Miller
AMXSY-CS, P. Beavers
C. Cairns
D. Frederick
AMXSY-G, J. Kramar
G. Comstock
E. Christman
L. Kravitz
AMXSY-GA, W. Brocks
AMXSY-J, A. LaGrange
AMXSY-L, J. McCarthy
AMXSY-P, J. Cullum
AMXSY-RA, R. Scungio
M. Smith

3 Dir, USABRL
ATTN: SLCBR-SE-P, W. Baker
J. Collins
E. Laurie

5 Cdr, USATECOM
ATTN: AMSTE-CG
AMSTE-TA-L, A. Yankolonis
N. Harrington
S. Grill
AMSTE-TC-C, R. Cozby

•U.S. GOVERNMENT PRINTING OFFICE: 1992-311-851/60031

USER EVALUATION SHEET/CHANGE OF ADDRESS

This laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers below will aid us in our efforts.

1. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

2. How, specifically, is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

3. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

4. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

BRL Report Number BRL-MR-3954 Division Symbol _____

Check here if desire to be removed from distribution list. ____

Check here for address change. ____

Current address: Organization _____
Address _____

DEPARTMENT OF THE ARMY
Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

OFFICIAL BUSINESS

BUSINESS REPLY MAIL
FIRST CLASS PERMIT No 0001, APG, MD

Postage will be paid by addressee.

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES